L Number	Hits	Search Text	DB	Time stamp
1	1	717.clas. and (harvard adj architecture)	USPAT	2004/08/13 09:29
2	519	harvard adj architecture	USPAT	2004/08/13 09:29
3	3	(harvard adj architecture) and (Java adj virtual adj machine)	USPAT	2004/08/13 09:30
-	167	(717/148).CCLS.	USPAT; US-PGPUB	2004/08/13 09:27
-	12	((717/148).CCLS.) and macro$	USPAT; US-PGPUB	2004/08/09 11:04
-	65	((717/148).CCLS.) and (repeat$ or repetitive$)	USPAT; US-PGPUB	2004/08/09 11:35
-	65	(((717/148).CCLS.) and (repeat$ or repetitive$)) and (cod$ ir instruction$)	USPAT; US-PGPUB	2004/08/09 11:36
-	47	(((717/148).CCLS.) and (repeat$ or repetitive$)) and sequenc$	USPAT; US-PGPUB	2004/08/09 14:16
-	2	717/148.ccls. and (repeat$ near (instruction$ or sequenc$))	USPAT	2004/08/09 15:09

01679616 SUPPLIER NUMBER: 15313858 (THIS IS THE FULL TEXT)

Clarification concerning modularization and McCabe's cyclomatic complexity. (Technical Correspondence) (Technical)

Feghali, Issa; Watson, Arthur H.; Henderson-Sellers, B.; Tegarden, David

Communications of the ACM, v37, n4, p91(4)
April, 1994
DOCUMENT TYPE: Technical ISSN: 0001-0782 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 2487 LINE COUNT: 00201

ABSTRACT: This correspondence deals with a prior article wherein Henderson-Sellers postulates modifications to the widely known cyclomatic complexity scheme originally published by McCabe. While Henderson-Sellers' conclusions are not entirely rebuffed, a useful modified relationship between his formulae and McCabe and Butler's integration complexity measure is demonstrated. Henderson-Sellers responds that his scheme considers modularization only by the division of an application into modules; doing away with duplicate coding is not addressed as a consequence.

TEXT:

The December 1992 issue of Communications contains a technical correspondence by Brian Henderson-Sellers (pp. 17-19) that misinterprets McCabe's cyclomatic complexity. While raising some interesting issues, the article contradicts McCabe's "landmark" publication [5] and what has become a national [6] and international standard in software measurement.

Since such misinterpretation may confuse a large number of programmers who have applied of McCabe's metric manually or through the use of automated tool, we feel this clarifying response is warranted. We will point out the analytical flaws behind Henderson-Sellers' proposed modification of the cyclomatic complexity formula, and then relate the modification to the integration complexity measure of McCabe and Butler [7].

Background

Computer scientists typically represent programs by flow graphs in which nodes correspond to sequential statements and edges to control flow. Graph theory [1] defines the cyclomatic number of a graph with n nodes, e edges, and p strongly connected components as e - 1 + p, which represents the number of fundamental cycles of the graph. McCabe built upon this theoretical foundation to define the cyclomatic complexity, v(G), of a program flow graph. Since the component flow graphs corresponding to program modules are not strongly connected, McCabe added an extra edge from the exit node to the entry node of each component module graph, resulting in the formula v(G) = e - n + 2p. McCabe showed that for each component module graph, v(G) is the cardinality of a basis set of paths through the module in the vector space of edge incidence. He then proposed the structured testing [6] methodology, according to which each module should be tested through a basis set of paths.

Correspondence in Question

Henderson-Sellers was primarily responding to a criticism of cyclomatic complexity, that since modularization makes programs "less complex," cyclomatic complexity should not increase with modularization. Henderson-Sellers proposed a modification of the definition of cyclomatic complexity, which coincides with McCabe's for single components but does not increase with modularization, presenting it as the correction of a simple algebraic error. Henderson-Sellers' formula, v(G) = e - n + p + 1, was presented as if adding a single edge to a multicomponent program flow graph would make each component strongly connected, which it would not. He showed that the modified formula is invariant if the code for a module that is called exactly once is expanded in-line. However, Henderson-Sellers' position is not supported by the fact that complexity dramatically increases if the expanded module was called more than once.

It is perfectly reasonable for the cyclomatic complexity of a program to increase with brute-force modularization. Each new module can be called

independently and has its own interface specification, which must be tested. As explained by McCabe [6], "this is quite different from a situation where A's code and B's code would be embedded within M." The increase in cyclomatic complextiy due to modularization (one test per module) is only significant for systems with simple modules. For such systems, modularization is in fact a detriment.

Contrary to Henderson-Sellers' assumption, cyclomatic complexity as given by McCabe's formual does not necessarily increase with modularization. In fact, cyclomatic complexity of a program can significantly decrease by selective modularization. When a section of code that appears more than once is modularized, the program's cyclomatic complexity is generally decreased. Thus, cyclomatic complexity has the desirable property of promoting modularization which eliminates code redundancy and increases module reuse.

Cyclomatic complexity also encourages modularization for reasons other than reducing the total cyclomatic complexity of a program. McCabe [5] suggested limiting the cyclomatic complexity of individual modules, since the interaction between the control structures in a complex module makes it more difficult to understand and test. Empirical studies [9] have supported McCabe's suggestion, showing a significant correlation between modules with high cyclomatic complexity and frequency of software errors. Thus, module-level cyclomatic complexity guides modularization based on reliability considerations.

We now consider the relationship between Henderson-Sellers' modified cyclomatic complexity formula and the integration complexity measure of McCabe and Butler [7].

McCabe and Butler defined the module design complexity, iv(G), to be the cyclomatic complexity of a module after a set of reduction rules have been applied. These rules preserve only the control structures that interact with module calls. McCabe and Butler defined the integration complexity, [S.sub.1], of a program to be [sigma](iv(G)) - p + 1, and suggested an integration testing methodology that requires [S.sub.1] tests. Henderson-Sellers' formula is [sigma](v(G)) - p + 1, which is the [S.sub.1] formula with iv(G) replaced by v(G). This formula could be viewed as an integration complexity measure without design reduction. It also gives the cardinality of a basis set of paths through an entire program in the vector space of edge incidence over all edges in all modules of the program.

Conclusion

McCabe's cyclomatic complexity is a useful measure for guiding modularization. The modification proposed by Henderson-Sellers is neither accurate nor necessary. However, we have shown an interesting relationship between his modified formula and the integration complexity measure of McCabe and Butler.

Response

Recent discussions highlight areas of confusion in both the theoretical basis of the cyclomatic complexity of McCabe [5] and its application to more than a single, connected flowgraph. In my technical correspondence

I introduced a revised equation for quantifying "cyclomatic complexity" which Feghali and Watson showed to be equivalent to the "integration complexity" [7] but with the module design complexity, iv(G), replaced by V(G). As such, this new formula, as pointed out, represents the "integration complexity without design reduction" and "gives the cardinality of a basis set of paths through an entire program...."

Cyclomatic Complexity

McCabe's [5] cyclomatic complexity, V(G), is a measure in widespread use for assessing the control complexity in a program. It aims to provide a basis set for constructing a testing program as well as identifying sub-programs which might be regarded as "overly complex." A basis set comprises a number of linearly independent (LI) paths through the program such that all other paths can be constructed from members of the basis set by using a vector-based approach.

With respect to the extension of V(G) to more than one component, there appear to be two options. Assuming that we are trying to describe a measurement effectively for th connected graph of Figure 1b but expressed in terms of the three disjoint components of Figure 1a, then the argument there is easily extended to give, inductively,

[V.sub.LI](G) = e - n + p + 1 (1)

This is the form proposed and relabeled here with a subscript LI since this metric is strongly related to the cycle rank of the graph [4].

Feghali and Watson point out that a different approach was taken by McCabe [5] (although apparently rescinded in McCabe and Butler [7](1)). He argued that each of the p components needs to be converted into a strongly connected graph, thus giving p graphs, themselves strongly connected, but not connected to each other. Here V(G) is given by

V(G) = e - n + 2p (2)

Thus, each component is treated independently and the value given by Equation (2) is relevant to each component but not really to the integrated system. Indeed, Shepperd [8] noted that this means a program with several subroutines is "treated as unconnected components within the control graph" which has "the bizarre result of increasing overall complexity if a program is divided into more, presumably simpler, modules." Each additional subprogram isolated from the rest of the program increases the value of V(G) by 1. Feghali and Watson interpret this additional complexity upon "brute-force modularization" as a test for the interface, rather than an addition to the t est path basis, which is the focus of [V.sub.LI](G) and the discussion here.

This concern led to Equation (1) which has the following properties.

* Modularization (prior to removal of repetitive code) has no effect on [V.sub.LI](G).

* The value of [V.sub.LI](G) for the full program is equal to the total number of decisions, D, plus one.

* The value of [V.sub.LI](G) is unchanged when subroutines are merged back into the program either by nesting or sequence. This ties in with the argument that the testing procedures (for the present limited to single calls to single entry, single exit components) are unchanged by modularization.

* The value of [V.sub.LI](G) provides a basis set for testing.

The relationship between the whole and the sum of the parts is also different. McCabe [5] shows that V(G) = [sigma]V([G.sub.i]) (3) whereas my technical correspondence deduces that [V.sub.LI](G) = [sigma][V.sub.LI]([G.sub.i]) + 1 - p (4) a form also supported in [2,3]. The difference here is that Fenton and Kaposi [3] are considering that summing components means merging them (by sequence or iteration); whereas the McCabe approach (Equation (3)) sums without merging. Thus, the focus is at the module level, where the most appropriate formula is [V.sub.LI](G) = V(G) = e - n + 2. The former retains an interpretation with respect to testing paths at the program level, but cannot be used for psychological complexity, since it would suggest that the "complexity" of the whole was less than the summed "complexities" of the parts. On the other hand, the McCabe metric, V(G), remains a stronger candidate for use as a full complexity measure since its value does not decrease when parts are summed to the whole.

My technical correspondence only considered modularization in the sense of dividing a program into modules and did not address the consequence of the second reason for modularization: the elimination of duplicate code chunks. Based on Equations (1) and (4), the [V.sub.LI](G) metric is extended to this case (extension to multiple entry and multiple exit modules is omitted here as a result of space constraints and is to be found in [4]). Since there are u duplicate chunks, the modularized program has one main routine which has u nodes and u - 1 edges, and u other components (viz., p = u + 1) (see Figure 2b). The value of [V.sub.LI](G) is unchanged (cf., the altered value of V(G). The concept that these p - 1 components are duplicates and can be replaced by a single module (Figure 2c) is introduced. Now there is the main routine plus only one other module: that of the duplicated chunk. This has e' edges and n' nodes. For the main module, [V.sub.LI]([G.sub.1]) = (u - 1) - u + 2 = 1 and for the submodule, [V.sub.LI]([G.sub.2]) = e' - n' + 2. Hence, from Equation (4) [V.sub.LI](G) = [V.sub.LI]([G.sub.1]) + [V.sub.LI]([G.sub.2]) + 1 - 2 = (e' - n') + 2 (5)

Since the value of the unmodularized code is given by [V.sub.LI](G) = u(e' - n') + u + 1 [4], it can be seen that the modularization to deal with each chunk of repeated code gives a reduction in the value of the cyclomatic complexity of (ue' - un' + u + 1) - (e' - n' + 2) = (u - 1)(e' -

n' + 1) = (u - 1)(v - 1)

where v is the complexity of the module being extracted. This result is identical to that given by Shepperd [8], and can thus be summed similarly over all of k repeated modules. This reduction also reflects the smaller cyclic basis so that the test strategy avoids duplicate testing of the duplicate code chunks.

In other words, as noted by Feghali and Watson, yet consistently with my technical correspondence, the cyclomatic complexity of a program containing duplicate chunks explicitly embedded in the main program (Figure 2a) is greater than the modularized version where the duplication is removed from the graph depicting the program (Figure 2c). Additional CALLs to the same subprogram engender no extra structural complexity since no additional test paths are created. Thus, Equation (4) holds for subroutines with multiple CALLs. However, it should be noted that such multiple CALLs may make a program harder to comphrehend and debug (a topic which cannot be addressed by the structural cyclomatic complexity measures discussed here). Feghali and Watson's comment on the dramatic increase of "complexity" in this situation is relevant to cognitive complexity but not structural complexity or the testing paths approach.

Discussion

The focus of both the measures V(G) and [V.sub.LI](G) is that of structural complexity. They can be related (with varying degrees of success) to the number of decisions and the provision of a basis set of testing paths. One different relates to whether V(G) = [sigma]V([G.sub.i]), which is true for McCabe's V(G) but not for [V.sub.LI]([G.sub.i]); although the latter retains consistency with the notion of a basis set. At the system level, [V.sub.LI](G) maintains a description of the basis set with regard to a testing methodology and complements the McCabe and Butler [7] notion of integration complexity, iv(G). The connection, as Feghali and Watson correctly note, is that [V.sub.LI](G) provides a test-paths view equivalent to this integration complexity without design reduction, whereby decision nodes not involved in subprogram CALLs are ignored. Therefore, the McCabe and Butler [7] formula for integration complexity, [S.sub.1] = [sigma]iv(G) + 1 - p is closely paralleled by Equation (4) for [V.sub.LI](G).

References

[1.] Berge, C. Graphs and Hypergraphs. North-Holland, Amsterdam, The Netherlands, 1973.

[2.] Evangelist, M. An analysis of control flow complexity. In Proceedings of COMPSAC 84. IEEE, 1984, pp. 388-396.

[3.] Fenton, N.E. and Kaposi, A.A. Metrics and software structure. Inf. Software Technol. 29 6 (Jun. 1987), 301-320.

[4.] Henderson-Sellers, B. and Tegarden, D. The application of cyclomatic complexity to multiple entry/exit modules. Centre for Information Technology Research Report No. 60, University of New South Wales, Sydney, Australia, 1993.

[5.] McCabe, T. A complexity measure. IEEE Trans. Software Eng., 2 4 (Apr. 1976), 308-320.

[6.] McCabe, T. Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric. NBS Special Publication 1982, 500-99.

[7.] McCabe, T.J. and Butler, C.W. Design complexity measurement and testing. Comm. ACM 32 12 (Dec. 1989), 1415-1425.

[8.] Shepperd, M. A critique of cyclomatic complexity as a software metric. Softw. Engi. J. 3, 2 (Mar. 1988), 30-36.

[9.] Walsh, T. A Software Reliability Study Using a Complexity Measure. In Proceedings of the National Computer Conference. AFIPS, N.Y., 1979, pp. 761-768.

SPECIAL FEATURES: illustration; chart
DESCRIPTORS: Modularity; Software Design; Mathematics of Computing; Software engineering; Theoretical Research
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: CD File 275

database courses at UCLA Extension and is the author of CA-Clipper 5...
...s degree in computer science from UCLA. His current interests lie in the area of Internet database **development** using the World Wide Web. (310)798-3985, Compuserve 73317,646, AMULET BBS (310)374-6999, Web...

...COMPANY NAMES: **Products**
DESCRIPTORS: **Product** Support...

... **Product** Tutorial

12/5,K/3 **(Item 3 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)

01679616 SUPPLIER NUMBER: 15313858 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Clarification concerning modularization and McCabe's cyclomatic complexity. (Technical Correspondence) (Technical)
Feghali, Issa; Watson, Arthur H.; Henderson-Sellers, B.; Tegarden, David
Communications of the ACM, v37, n4, p91(4)
April, 1994
DOCUMENT TYPE: Technical ISSN: 0001-0782 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 2487 LINE COUNT: 00201

ABSTRACT: This correspondence deals with a prior article wherein Henderson-Sellers postulates modifications to the widely known cyclomatic complexity scheme originally published by McCabe. While Henderson-Sellers' conclusions are not entirely rebuffed, a useful modified relationship between his formulae and McCabe and Butler's integration complexity measure is demonstrated. Henderson-Sellers responds that his scheme considers modularization only by the division of an application into modules; doing away with **duplicate** coding is not addressed as a consequence.

SPECIAL FEATURES: illustration; chart
DESCRIPTORS: Modularity; Software Design; Mathematics of Computing; Software engineering; Theoretical Research
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: CD File 275

...ABSTRACT: that his scheme considers modularization only by the division of an application into modules; doing away with **duplicate** coding is not addressed as a consequence.
... and Butler [7].
Background
Computer scientists typically represent programs by flow graphs in which nodes correspond to **sequential statements** and edges to control flow. Graph theory [1] defines the cyclomatic number of a graph with n...

...testing program as well as identifying sub-programs which might be regarded as "overly complex." A basis **set** comprises a **number** of linearly independent (LI) paths through the program such that all other paths can be constructed from...into modules and did not address the consequence of the second reason for modularization: the elimination of **duplicate** code chunks. Based on Equations (1) and (4), the [V.sub.LI](G) metric is extended to...

...as a result of space constraints and is to be found in [4]). Since there are u **duplicate** chunks, the modularized program has one main routine which has u nodes and u - 1 edges, and...

...is unchanged (cf., the altered value of V(G). The concept that these p - 1 components are **duplicates** and can be replaced by a single module (Figure 2c) is introduced. Now there is the main routine plus only one other module: that of the **duplicated** chunk. This has e' edges and n' nodes. For the main module, [V.sub.LI]([G.sub...

...n') + u + 1 [4], it can be seen that the modularization to deal with each chunk of **repeated** code gives a reduction in the value of the cyclomatic complexity of (ue' - un' + u + 1) - (e...

...identical to that given by Shepperd [8], and can thus be summed similarly over all of k **repeated** modules. This reduction also reflects the smaller cyclic basis so that the test strategy avoids **duplicate** testing of the **duplicate** code chunks.

In other words, as noted by Feghali and Watson, yet consistently with my technical correspondence, the cyclomatic complexity of a program containing **duplicate** chunks explicitly embedded in the main program (Figure 2a) is greater than the modularized version where the **duplication** is removed from the graph depicting the program (Figure 2c). Additional CALLs to the same subprogram engender no extra structural complexity since no additional test paths are **created** . Thus, Equation (4) holds for subroutines with multiple CALLs. However, it should be noted that such multiple...

12/5,K/4 (Item 4 from file: 275)

01521909 SUPPLIER NUMBER: 12353479 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Parallel database systems: the future of high performance database systems. (Technical)
DeWitt, David; Gray, Jim
Communications of the ACM, v35, n6, p85(14)
June, 1992
DOCUMENT TYPE: Technical ISSN: 0001-0782 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 10200 LINE COUNT: 00840

ABSTRACT: Successful parallel database systems are based on conventional processors, memories and disks; the systems realize the potential of fast-cheap commodity disks, processors and memories expected in the future. The consensus parallel and distributed database system architecture is based on a shared-nothing hardware design in which processors communicate by sending messages through an interconnection network. Tuples of each relation in the database are partitioned, or declustered, across disk storage units attached directly to each processor. This allows several processors to scan large relations in parallel without requiring exotic I/O devices. The architectural concepts used in parallel database systems are described, and the unique features of the Teradata, Tandem, Bubba and Gamma systems are examined.

SPECIAL FEATURES: illustration; table; chart; graph
DESCRIPTORS: Database Design; Research and **Development** ; Processor Architecture; Data Flow Diagrams; Partitioned Files; Parallel processing; DBMS; Connectivity
SIC CODES: 3571 Electronic computers
FILE SEGMENT: AI File 88

... parallel database systems. Over the last decade Teradata, Tandem, and a host of startup companies have successfully **developed** and marketed highly parallel machines.

Why have parallel database systems become more than a research curiosity? One...

...suited to parallel execution; they consist of uniform operations applied to uniform streams of data. Each operator **produces** a new relation, so the operators can be composed into highly parallel dataflow graphs. By streaming the...

...several research projects. This design is now used by Teradata, Tandem, NCR, Oracle-nCUBE, and several other **products** currently under **development** . The research community has also embraced this shared-nothing

dataflow architecture in systems like Arbre, Bubba, and...evolutionary step toward the shared-nothing design. Partitioning a shared-nothing design. Partitioning a shared-memory system **creates** many of the skew and load balancing problems faced by a shared-nothing machine; ...this protocol, but they all end up exchanging reservation messages and exchanging large physical data pages. This **creates** processor interference and delays. It **creates** heavy traffic on the shared interconnection network.

For shared database applications, the shared-disk approach is much...

...the tuples in a relation have the same set of attributes (fields in COBOL terminology).

Relations are **created** , updated, and queried by writing SQL statements. These statements are syntactic sugar for a simple set of...

...from the relational algebra. Select-project, here called scan, is the simplest and most common operator--it **produces** a row-and-column subset of a relational table. A scan of relation R using predicate P and attribute list L **produces** a relational data stream as output. The scan reads each tuple, t, of R and applies the...

...to a sort operator that will reorder the tuples based on an attribute sort criteria, optionally eliminating **duplicates** . SQL defines several aggregate operators to summarize attributes into a single value, for example, taking the sum...

...operator (here called join). The join operator composes two relations, A and B, on some attribute to **produce** a third relation. For each tuple, ta, in A, the join finds all tuples, tb, in B...high-level direction of the SQL dataflow.

The SQL data model was originally proposed to improve programmer **productivity** by offering a nonprocedural database language. Data independence was an additional benefit; since the programs do not...

...distributed partitioning criteria. Bubba uses this concept by considering the access frequency (heat) of each tuple when **creating** partitions of a relation; the goal being to balance the frequency with which each partition is accessed...data streams instead of writing new parallel operators (programs). This approach enables the use of unmodified, existing **sequential routines** to execute the relational operators in parallel. Each relational operator has a set of input ports on...

...implemented as three scan operators that send their output to a common merge operator. The merge operator **produces** a single output data stream to the application or to the next relational operator. The parallel query executor **creates** the three scan processes shown in Figure 8 and directs them to take their inputs from three...

...into several independent streams. A split operator is used to partition or replicate the stream of tuples **produced** by a relational operator. A split operator defines a mapping from one or more attribute values of...

...nothing machine architectures.

The split operator in Table 1 is just an example. Other split operators might **duplicate** the input stream, or partition it round-robin, or partition it by hash. The partitioning function can...

...example of these improved algorithms.

Recall that the join operator combines two relations, A and B, to **produce** a third relation containing all tuple pairs from A and B with matching attribute values. The conventional...In case of data skew, some sort partitions may be much larger than others. This in turn **creates** execution skew and limits speedup and scaleup. These skew problems do not appear in centralized sort-merge...the hardware sorter clearly contradict the thesis that special-purpose hardware is not a good investment of **development** resources. Time will tell whether these special-purpose components offer better price performance or peak performance than...

...Other parallel database system prototypes include XPRS [30], Volcano [14], Arbre [21], and the PERSIST project under **development** at IBM Research Labs in Hawthorne and Almaden. While both Volcano and XPRS are implemented on shared...

...in peak performance and in price/performance [13].
The NCR Corporation has announced the 3600 and 3700 **product** lines that employ shared-nothing architectures running System VR4 of Unix on Intel 486 and 586 processors. The interconnection network for the 3600 **product** line uses an enhanced Y-Net licensed from Teradata while the 3700 is based on a new multistage interconnection network being **developed** jointly by NCR and Teradata. Two software offerings have been announced. The first, a port of the...versioning mechanism that gives readers a consistent (old) version of the database while updators are allowed to **create** newer versions of objects. Other, perhaps better, solutions for this problem may also exist.
Priority scheduling is...

...disks. The resulting output would suggest a partitioning strategy for each relation plus the indices to be **created** on each relation. Steps in this direction are beginning to appear.
Current algorithms partition relations using the...

...longitude or latitude. Partitioning on longitude allows selections for a longitude range to be localized to a **limited number** of nodes, selections on latitude must be sent to all the nodes. While this is acceptable in...

...essential that the data be available while the utilities are operating. In the SQL world, typical utilities **create** indices, add or drop attributes, add constraints, and physically reorganize the data, changing its clustering.
One unexplored...computational and I/O resources available only from a parallel architecture.
While the successes of both commercial **products** and prototypes demonstrate the viability of highly parallel database machines, several research issues remain unsolved including techniques...

...Tanaka, H., and Moto-oka, T. Application of hash to data base machine and its architecture. New **Generation** Computing 1, 1 (1983).
[19]. Kitsuregawa, M., Yang, W., and Fushimi, S. Evaluation of 18-stage pipeline...

...DESCRIPTORS: Research and **Development** ;

12/5,K/5 (Item 5 from file: 275)

01516565 SUPPLIER NUMBER: 12180100 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Some random thoughts. (programming random-number generators **for various applications) (Software Exploratorium) (column) (Tutorial)**
Bentley, Jon
UNIX Review, v10, n6, p71(7)
June, 1992
DOCUMENT TYPE: Tutorial ISSN: 0742-3136 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 3433 LINE COUNT: 00290

ABSTRACT: Random-number **generators** are required for various scientific applications; approaches to finding approximate solutions to very large traveling-salesman problems (TSP) are described. Researchers often want to compare programs on common data sets; the problems can be **generated** using a portable random-number **generator** rather than by shipping around a library of problems. A library would describe a 10,000-city problem by specifying each point by two seven-digit numbers, using a total of

(lprand()*1e9 + lprand()) * 1.e-18
11. A simple experiment showed that **generating** a random integer with the expression fprand()*n gave much more appropriate results for fortune. To explore...

...in the file. I found that for every seed less than 100,000, whenever the sixth integer **generated** is congruent to 0 modulo 6, the ninth integer is congruent to 0 modulo 9 (and thus...

DESCRIPTORS: Random Number **Generation** ; ...

...Program **Development** Techniques

12/5,K/6 **(Item 6 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)

01511124 SUPPLIER NUMBER: 12063134 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Notes on implementing sets in Prolog.
Munakata, Toshinori
Communications of the ACM, v35, n3, p112(9)
March, 1992
ISSN: 0001-0782 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 6384 LINE COUNT: 00495

ABSTRACT: Prolog, which stands for Programming Logic, was **developed** in France during the 1970s and was chosen as the official programming language for Japan's Fifth **Generation** Computer Project in 1981. The artificial intelligence programming language is based on predicate logic and has a simple and powerful declarative symbolic language base. Prolog has a wide range of applications that include expert systems, relational data bases, image processing, VLSI circuit analysis, symbolic algebra, compiler writing and natural language processing. A comprehensive detailing of how to implement and manipulate sets within Prolog is presented.

SPECIAL FEATURES: illustration; table; program
DESCRIPTORS: Program **Development** Techniques; Data Structures; Software Engineering; PROLOG; Programming Language; Artificial intelligence
FILE SEGMENT: AI File 88

ABSTRACT: Prolog, which stands for Programming Logic, was **developed** in France during the 1970s and was chosen as the official programming language for Japan's Fifth **Generation** Computer Project in 1981. The artificial intelligence programming language is based on predicate logic and has a...

TEXT:
...brought to world attention when chosen as the official language for the well-publicized, Japan's Fifth **Generation** Computer Project, in 1981. Currently, Prolog is one of the two major artifical intelligence languages, the other...

...relational databases, and more recently, image processing. Prolog is sometimes used in an early stage of system **development** for quick implementation. A successful Prolog system that is likely to be used extensively is sometimes rewritten...
... extensions include more powerful facilities (e.g., Prolog III CLP--Constraint Logic Programming [2, 8, 9], and **development** of concurrent versions of Prolog (e.g., [15] Shared Prolog [4], Parlog [14]). Prolog has also been implemented as hardware for faster computation. They include Japan's Fifth **Generation** Computer Project and VLSI Prolog chips **developed** in Europe. In the fall of 1990, a small group of people in Japan organized the Japan...

...when represented as a list or an array. Another difference is that a set does not include **duplicated** elements such as, (7, 7, 8, 9). (If there are, they are interpreted as representing the same **Generate** explicit

representation always 3b: **Generate** explicit representation as needed 3c: Direct manipulation on implicit forms

Method 4: Other Methods

Method 1 which...of cities in 'usa', or the set of large cities where each population is greater than a **certain number**, and so on.

Basically we will discuss two types of approaches (Methods 2a and 2b) to perform...

...This procedure first places the elements in our own temporary queue, recollects them into a list, then **generates** a new fact by assertz [3]. Here we assume that the set to be built is not...X] Rest], collect(Rest)).

When this predicate is invoked by build set(data, a, set), it will **generate** set(a, [2, 3, 5]) from data(a, 2), data(a, 3) and data(a, 5).

Undoing...

...want to reverse the build set process. For example, given

set(c, [4, 5]).

we want to **generate** :

data(c, 4). data(c, 5).

We can use the following:

build data(S);- set(S, L), **generate** (S, L). **generate** (S, [ ]):- !. **generate** (S, [X] L1]):- assertz(data(S, X)), **generate** (S, L1).

The Prolog question

?- build data(c).

will yield the desired result for the prior example...

...be invoked by ?- build data (data, c, set). built[underscore]data(P, S, Q) :- Q(S,L), **generates** (S,L). **generates** (S,[]):-!. **generates** (S, [X] LI]):- assertz(P, (S,X).).

Method 2b

We will now introduce the approach which operates...

...accumulator to count the number of elements.

In Table 2, acc(Acc) is a fact to be **created** during this process, and its argument, Acc, is the temporary accumulator. Acc is initialized to 0 when...

...The execution then goes to the second clause of 'count', where the final processing is performed.

To **generate** a new set C as the union of sets A and B such as, data(c,5...

...discussed.

Prolog procedures can be used to described the characteristics of the elements of a set. A **procedure** is a **sequence** of clauses whose heads have the same predicate. Certain operations on the arguments of procedure may actually...

...sets than the explicit mehod, especially for large sets. If explicit elements are necessary, let the computer **generate** those elements from given characteristics. Second, the explicit method is clearly impossible for an infinite set. There...

...We will discuss three types of approaches for manipulating sets using the implicit method. Method 3a first **generates** all the explicit elements of the ...Method 1 or 2, that is, we reduce the problem to 'explicit' methods. Method 3b is to **generate** explicit elements whenever necessary and perform operations. Method 3c is to deal with the characteristics directly without **generating** explicit elements.

Method 3a. Reducing the Problem to Method 1 or 2 by **Generating** all the Elements

This is a two-step method. In Step 1, we **generate** all the explicit elements of sets for given characteristics. In Step 2, we perform set operations on these explicit elements by using Method 1 or 2. If **generating** all the elements is not possible for some reason, obviously this method cannot be used.

Example. The **set** of Fibonacci **number** in explicit form. Using the previously defined procedure fib(N, FN), we can **generate** the set of Fibbonacci numbers by explicitly writing out the elements. The following procedure fibset(N, S...

...city[underscore]in(large[underscore]cities, new york).
city[underscore]in(large[underscore],cities, london).

We can **generate** this type of explicit form of facts from the implicit form by first placing the body of the implicit form, then asserting the head of the implicit form as a fact. For example, to **generate** the facts for large-cities from the implicit form:

city[underscore]in(large cities[underscore], City) :- city (, City, Pop), Pop > 5000.

Invoke the following by?- **generate** .

generate :- city(-, City, Pop), Pop > 5000, assertz (city in(large cities, City)).

After obtaining the explicit elements, we...

...be applied. Assuming that there are n facts in the database, O(n) comparisons are necessary to **generate** the explicit form of elements.

Method 3b. **Generating** Explicit
Elements on Demands

Generating all the elements of sets for given characteristics as in Method 3a may or may not be...

...C; otherwise do not include it. To determine the union of A and B, we have to **generate** all the elements of A and B.

Method 3c. Direct Manipulation of
Sets in the Implicit Form

Performing operatins on implicitly represented sets without **generating** explicit elements is an attractive approach. Generally, however, it is harder to perform such set operations than...
...are possible.

Union and intersection using conjunctive and disjuntive subgoals. For given predicates A and B, to **generate** the union of A and B as C, assert a new rule in form of C :- A...

...mod 2) == 0.
odd[underscore]no(N) :- integer(N), (N mod 2) == 1.

Suppose we want to **generate** the union of even[underscore]no(N) and odd[underscore]no(N) as a new procedure called...B1, B2), conj[underscore]subgoal(B, B2).

Note that there we only deal with the predicates without **generating** specific elements to determine a subset relation. Procedure disj[underscore]subset(A, B) for a rule involving by Examples. Springer-Verlag, Berlin, 1988.

[7] Cohen, J. A view of the origins and **development** of Prolog. Commun. ACM, 31, 1 (Jan. 1988), 26-36.

[8] Cohen J. Constraint logic programming languages...

DESCRIPTORS: Program **Development** Techniques...

12/5,K/7 (Item 7 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01503633 SUPPLIER NUMBER: 11936850 (USE FORMAT 7 OR 9 FOR FULL TEXT)
The device driver as state machine. (logical constructs)
Nelson, Thomas
C Users Journal, v10, n3, p41(1)
March, 1992
ISSN: 0898-9788 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT
WORD COUNT: 4752 LINE COUNT: 00356

DESCRIPTORS: Device Driver; Code **Generator** ; Programming; Interface; Applications; Program **Development** Techniques; Operating System

Enhancement
FILE SEGMENT: CD File 275

TEXT:
...of abstraction between the user and the controlled device. It also isolates device-dependent code within a **limited number** of functions.
... up to 64Kb in length, forcing all code and data into a single segment. Some compilers can **produce** a tiny model .COM file that satisfies these requirements. However, a device driver is a binary image...

...going to support compiled C code, the startup procedures must occur here also. Your startup code must **duplicate** as closely as possible the environment **created** by the compiler's normal startup code. The ... separate stack segment, however, and will complain about it. This is one warning you can ignore. Although **creating** a separate stack for the driver is not strictly necessary, supporting a stack-intensive language like C... driver initialization code, attempting to drop init 0 is probably more trouble than it's worth. To **repeat** , you're programming a driver in C for C's relative ease of use, not primarily to...

...open the device instead of your disk file and you will get some unexpected results. To illustrate, **create** a file called lpt1 with your text editor. Try writing some text to it using the DOS...When the machine reaches the next state, it waits until it receives another event or command, then **repeats** the cycle.
The State Table In C
To illustrate the basic concepts of device control using a state machine, I **developed** a simple example that controls a hypothetical tape backup unit. To build the state table, start with...

...single array of structures. Although easy to understand, such an array will mean some degree of data **repetition** . I have instead used a series of cascading or multi-level tables connected by pointers, resulting in...
...TABS, the basic data element, to **create** an event table (Listing 4). The last member of an event table must be the macro END...

...initialized data in tables significantly reduces your coding effort. Without them, your code would be littered with **logical** branching **statements** .
The IOCTL Interface
DOS device drivers make available at least two command code routines to control the...

...write(), respectively (Listing 2). A **developer** can use these functions for any purpose since nothing is specified other than the calling protocol.
As...EXE test bed, using compile-time #defines. Leave the test bed code in place, inactivated, when you **produce** the finished driver. When you make changes and retest, simply unpack the test bed code.
Testing your...

...driver still needs to be tested when linked into the DOS driver chain. To prepare for this, **create** a bootable floppy complete with a dummy config.sys that includes a command to load your test...

...DESCRIPTORS: Code **Generator** ; ...

...Program **Development** Techniques

12/5,K/8 (Item 8 from file: 275)

01496429 SUPPLIER NUMBER: 11719810 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Adapting formal testing techniques for Windows applications. (Tutorial)
Bilson, Gretchen

Microsoft Systems Journal, v7, n1, p77(20)
Jan-Feb, 1992
DOCUMENT TYPE: Tutorial ISSN: 0889-9932 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 6070 LINE COUNT: 00479

ABSTRACT: Testing Microsoft Windows applications is time-consuming and is not much fun. Still, **developers** need to be sure their applications will run for a certain amount of time without unrecoverable errors. One testing approach, called branch coverage, runs every branch of a conditional statement. There are two basic kinds of testing: functional testing and structural testing. With functional testing, an input is defined, the application is invoked, and the results are compared with the application's specifications. Structural testing, usually performed by an advanced programmer, considers such details of program structure as language, control flow and programming style. Code is given for Autotest, a capture/playback tool helpful for stress-testing a Windows application just before it is sent to **production** . Autotest will help locate subtle bugs that will reveal themselves during **repeated** program execution.

SPECIAL FEATURES: illustration; program; table; chart
DESCRIPTORS: Tutorial; Software Quality; Testing; Methods; Type-In Programs; Applications Programming
TRADE NAMES: Microsoft Windows (GUI)--Computer programs
FILE SEGMENT: CD File 275

ABSTRACT: Testing Microsoft Windows applications is time-consuming and is not much fun. Still, **developers** need to be sure their applications will run for a certain amount of time without unrecoverable errors...

...a capture/playback tool helpful for stress-testing a Windows application just before it is sent to **production** . Autotest will help locate subtle bugs that will reveal themselves during **repeated** program execution.

TEXT:

My favorite question to ask **developers** working in the Microsoft-" Windows" graphical environment is, "Would you get in a plane flown by an...

...consequences of disabled or malfunctioning software. Obviously, these applications must be rigorously tested before being released into **production** .

... free execution is still essential. You need to be confident that your application will run for a **certain number** of hours or days without unrecoverable errors. Many experienced Windows applications **developers** have performed what seemed like exhaustive tests on their system and released them into **production** , only to get a call later from an irate user complaining about system freezes. (I received several...

...a few times. Testing your Windows application can eat up 50 percent or more of your total **development** time-but it's time nobody likes to talk about. The pain is quickly forgotten once it...

...of nested case statements versus the traditional sort of program testing theorists had in mind when they **developed** their ideas. One difference seems to be the much heavier, line-by-line reliance a Windows application ...

...of you would argue that this is a mighty big assumption!) Compared to a Windows application under **development** , it is reasonable to assume that Windows itself is bug-free.

Testing Techniques

Testing techniques can be...

...testing techniques, is based on the selection of test paths through a program. A path is some **sequence** of **statements** that begins at an entry point, a junction (labeled statement), or a decision (conditional expression); includes zero...

...procedures very often have control flow structures similar to WinProcs. To simplify testing, Windows routines should be **developed** with a single-entry-single-exit structure wherever possible, because single-entry-multiple-exit routines make it...the application's source code. It's most useful as part of unit testing by the application **developer** . But because statistics show that path testing catches only 35 percent of all bugs (see Error Detection...

...theory is usually explained in terms of program implementation.

Let's look at the Microsoft Windows Software **Development** Kit (SDK) program GENERIC.C to illustrate these concepts. Windows can send input from a variety of sources to the WinProc. This input may be **generated** by the user from the keyboard or mouse, from a timer, by Windows as part of its...

...that classify the domains in this example. (I have ignored the domains that will result from the **creation** of the dialog box routine. The dialog box function is its own callback routine and has its...have coincidental correctness errors; that is, errors where the execution outcome is correct but the process that **generated** it is wrong.

Since multidimensional domains are difficult to visualize and hard to test without automated tools...

...domains as seen by the calling function and the called function.

In function interfaces, the output values **produced** by a function or subprocess are passed from the calling function to the called function. These output...code to validate parameters has been incorporated in the beta releases of Windows 3. 1. This prevents **developers** from passing invalid parameters to Windows functions, eliminating some GP faults and other UAES. Windows 3.1...

...With each testing theory discussed, even the simplest program may require thousands of test cases to be **generated** . It is impractical to apply these testing methods without the help of automated tools. For **developers** to test efficiently, automated tools that are engineered for a particular operating environment and language are needed...

...on the statistical model, is useful for stress-testing Windows applications just prior to their release into **production** . Autotest can help reveal memory-allocation problems such as incorrect memory flags, undiscarded objects, problems with errant pointers, and other subtle bugs that manifest themselves during **repeated** program execution.

This tool can greatly reduce the time required to detect these bugs. Autotest is also...

...any application's message loop.

Every test tool used to insert probes in application code or to **generate** traces can reveal information about the system. However, the more information a tool collects, the more the...on Windows hooks can be found in Chapter 6 of Jeffrey Richter's book, Windows 3: A **Developer** 's Guide (M&T Books, 199 1).

Tester calls the SetWindowsHook function to install the WH...

...coordinates. If a keyboard message is recorded, paramL and paramH hold the key scan codes and key **repeat** counts. When a mouse message is played back the mouse location is used to determine which window...STOPPLAY, (HANDLE)NULL, (HWND)NULL,

0, TraceOn, (LPSTR)NULL);

Finding Bugs with Autotest

I **created** the original version of Autotest (see Figure 23) to stress-test a suite of Windows applications I...

...stress tests. First I determined the minimum length of time an application had to be operational. I **generated** test cases with multiple instances of each application and exercised different paths within each instance, making sure...

...system had to be operational. I would gradually rerun the test cases at

faster speeds and then **repeat** the test with a group of different applications that had been already stress-tested alone. My objective...

...would be useful. More information on the Windows environment would be especially helpful if it could be **generated** without adding to the overhead of the tool. It would also be desirable to control the reporting ...

...Windows environment.

Although I've focused exclusively on program-based testing performed at the end of the **development** process, I strongly recommend integrating testing into your **development** plans from the very beginning of a project. It's like the oil filter commercial says, You can pay me now or pay me later." Preventing bugs early in the **development** cycle minimizes the time required to test at the end of a project.

Maybe your autopilot will...

12/5,K/11> **(Item 11 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2004 The Gale Group. All rts. reserv.

01372441 SUPPLIER NUMBER: 09452475 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Concurrent object-oriented programming. (includes related article on multicomputers)
Agha, Gul
Communications of the ACM, v33, n9, p125(17)
Sept, 1990
ISSN: 0001-0782 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 12260 LINE COUNT: 01002

ABSTRACT: Three significant trends in concurrent computing are increased use of interacting processes by individual users; the use of cost-effective workstation networks as the primary way to share resources and distribute them for problem solving; and multiprocessor technology, which delivers supercomputing power at a much lower price. Software engineering has moved toward data abstraction to promote program modularity, as shown by the increased acceptance of object-oriented programming. Concurrent object-oriented programming (COOP) provides a software foundation for concurrent computing on multiprocessors; the foundations and methodology of COOP are discussed. COOP includes a range of structures that can be used to express patterns of concurrent problem solving. The actor model as a framework for concurrent systems is discussed specifically.
CAPTIONS: A transactional structure: each request **generates** a unique response. (chart); A simple prime sieve. (chart); A recursive factorial computation. (chart)

SPECIAL FEATURES: illustration; chart
DESCRIPTORS: Object-Oriented Programming; Multiprocessing; Software Design; Modular Programming; Concurrent Programming; New Technique; Program **Development** Techniques
FILE SEGMENT: CD File 275

... the first object-oriented language, simulated a simple form of concurrency using coroutines on conventional architectures. Current **development** of concurrent object-oriented programming (COOP) is providing a solid software foundation for concurrent computing on multiprocessors. Future **generation** computing systems are likely to be based on the foundations being **developed** by this emerging software technology.

The goal of this article is to discuss the foundations and methodology...

...going projects in COOP.

It is important to note that the actor languages give special emphasis to **developing** flexible program structures which simplify reasoning about programs. By reasoning we do not narrowly restrict ourselves to...

creating and waiting for tasks, or managing task identifiers. The loop iteration variable serves as the task identifier...

...DESCRIPTORS: Program **Development** Techniques...

12/5,K/16 **(Item 16 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2004 The Gale Group. All rts. reserv.

01291373 SUPPLIER NUMBER: 07060804 (USE FORMAT 7 OR 9 FOR FULL TEXT)
More on data integrity with Advanced Revelation.
Gunther, John
Data Based Advisor, v7, n1, p36(3)
Jan, 1989
ISSN: 0740-5200 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 2683 LINE COUNT: 00213

ABSTRACT: Options which can be used to customize data entry prompts in Advanced Revelation windows are examined, with a focus on those that prevent invalid data from getting into files and those that help operators enter data efficiently and accurately. A useful and instructive data conversion routine is presented. The Default, Tab Stop, Edit Mask, and Max Length options are also discussed.

DESCRIPTORS: Programming; Tutorial; Data Integrity; Data Entry
SIC CODES: 7372 Prepackaged software
TRADE NAMES: Advanced Revelation (Data base management system)--Usage
FILE SEGMENT: CD File 275

... and displayed in windows using the terminology True/False, Yes/No, or On/Off.

Program 1: Conversion **routine** for Boolean **logical** values SUBROUTINE LOGIC$CNV (TYPE,IN.VALUE,DISPLAY.TYPE,OUT.VALUE) *User conversion routine to input and...John" or "1997", can be specified by typing it as the value of the Default option.

The **repeat** specification (") sets the default value of the field to whatever the field was in the previously displayed...

...This one's handy when your calculation involves other fields in the same record or you're **creating** a default routine that is only relevant within one file. Fields in other files won't be...

...subroutine ([subroutine.name]). It's similar to the formula method except that it's best used to **create** default specifications accessible to a number of files.

The third user-defined type is a CATALIST ...the Window.max length option.

Window.max length option

For fields in which the input is a **fixed** **number** of characters, such as serial numbers, social security numbers, etc., the Max Length prompt option is available...
...Max Length can be very useful.

Additional items

Here's a tip for advanced RBASIC programmers: AREV **generates** a runtime error whenever a program references a variable that hasn't yet been assigned. During the...

...shown as @ AN.

John Gunther owns a broad spectrum consulting firm with particular strengths in networking, applications **development** , and third-party support. He is also a consult for Abacus Computer Consulting of New York City...

12/5,K/17 **(Item 17 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)

number. Virtual fields would include Last Order...
...UNI-FILE's report writer is limited to a single file, you won't be able to **duplicate** on paper what you can see on the screen.

Now that I've thrown water on the...pull most novices through the learning curve unscathed. --Dick Ridington

Table:

Photo: With Alpha/three, you can **create** search criteria with a table mode or an equation mode. This flexibility is available in many of...

...status line for easy reference.

Photo: The Data Reporter's graphics package gives you the option of **generating** a statistical report on any data file.

Photo: DataPlus-86's data entry fields are predetermined. You can accept all field titles shown, accept some of them, or **create** your own.

Photo: The standard Flexifiler data-file definition screen asks that you verify the information before...

...command bar.

Photo: InfoStar Plus's rather simplistic main menu is typical of menus that can be **created** with the Starburst utility.

Photo: Palantir Filer uses screen-painted data entry forms. The Microsoft Windows menu...

12/5,K/19 **(Item 19 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)

01177490 SUPPLIER NUMBER: 04470961 (USE FORMAT 7 OR 9 FOR FULL TEXT)
A comprehensive guide to 1-2-3's macros.
Taylor, Jared
PC Magazine, v5, n18, p301(8)
Oct 28, 1986
ISSN: 0888-8507 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 4607 LINE COUNT: 00310

ABSTRACT: Macros are key-based commands that execute other commands. Program commands that are typed from the keyboard can be organized into macros. The program will run various commands when the user presses a key. Virtually any keyboard task that **repeats** itself on the program can be converted into a macro that performs the task effortlessly and instantly. Proficiency in the use of macros can be attained through rigorous practice and trial and error.
CAPTIONS: **Productivity** index. (table); Macro prompts. (program)

SPECIAL FEATURES: illustration; table; program
DESCRIPTORS: Macro; Personal Computers; Tutorial; Software Packages; Keys ; Keyboard; Assembly Language; Codes; Spreadsheet Software
FILE SEGMENT: CD File 275
...ABSTRACT: The program will run various commands when the user presses a key. Virtually any keyboard task that **repeats** itself on the program can be converted into a macro that performs the task effortlessly and instantly ...
... 2-3 recognizes them as macro instructions; and (3) actually run, or "invoke,' the instructions.

Let's **create** this simple macro. First, blank the 1-2-3 screen with
/wey
then move the cursor to cell C4. All **macro instruction sequences** must be, in 1-2-3 parlance, labels rather than numbers. In the case of our column...

...t forget to start the name with the apostrophe label prefix, because the backslash itself is the **repeating** label prefix. Without the apostrophe you'll get a string of W's instead of W.

Now...

...3 recognizes it as a macro. Leave the cursor in the cell with W in it

and **create** a range as follows
/rnlr<Enter>
This gives the range name W to the cell to the right--that is, the cell with the series of instructions in it. You can use /Range Name **Create** instead of /Range Name Label Right, but it's a good practice to put the range name...to the cell to its right. It forces you to label your macros and also lets you **create** a bunch of range names at a time.
Now your macro is ready to run. Hold down...
...left>), >(up>), >(pgdn>), etc., all work in macros the way you would expect them to.
Cell C6 **repeats** the instructions in C4, and all the rest of the macro is **repetition** . This macro will set the width of the current column to six, then move to the next...

...1-2-3 macro do something more than once. Rather than write new macro instructions for every **repetition** , you can make 1-2-3 go back and execute the same instructions over and over.
Get...in Figure 4. This macro prompts you twice: once for the column width and once for the **number** of columns to **set** to that width. After you give it that information, it will go off and do its work...do much, much more. You can write macros that draw graphs, do database operations, consolidate spreadsheets, and **create** custom menus that work just like 1-2-3's own built-in menus. There is no...

...lines long, and it could be rewritten onto a single line. 1-2-3 can be tremendously **productive** even if you don't use every @ function, and macros can be tremendously helpful even if they...

...Figure 2: A more elegant way to make a macro do something more than once--essentially, a **repeating** loop.
Photo: Figure 3: A macro that prompts for a column width and then formats columns accordingly...
CAPTIONS: **Productivity** index. (table); Macro prompts. (program)

12/5,K/20 (Item 20 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01176603 SUPPLIER NUMBER: 04366955 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Relational databases. (Software Review) (evaluation)
Dickinson, John
PC Magazine, v5, n12, p184(33)
June 24, 1986
DOCUMENT TYPE: evaluation ISSN: 0888-8507 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 21032 LINE COUNT: 01606

ABSTRACT: The 24 relational database managers evaluated in this section differ from programmable packages in that they can be used without having to master the complexities of programming, often making them easier to learn and operate, and sometimes allowing them to provide more power than programmable packages. They differ from flat-file database managers in being much more suitable for business applications, allowing users to enter common data for a number of records only once rather than having to **repeat** the information for each record. They also allow files to be connected in a manner that allows common data to be accessed by several records in one or more additional files. There is considerable variety within the relational non-programmable category, meaning a variety of different users are likely to find a package that will serve their particular needs.
CAPTIONS: Fact files are included on the 24 packages evaluated. (table); A table lists the features of each of the 24 packages evaluated. (table); Index: relational databases. (table); Editor's choice. (table)

SPECIAL FEATURES: illustration; photograph; table
DESCRIPTORS: Relational Data Base Management Systems; DBMS; Software Packages; Personal Computers; Evaluation; Data Management; Comparison

```
Set     Items   Description
S1    5634489   CREAT? OR GENERAT? OR PRODUC? OR DEVELOP?
S2    2294586   MACRO()INSTRUCTION? OR STATEMENT? OR OPERATION? OR PROCEDU-
            RE? OR ROUTINE?
S3       4260   (SEQUENCE? OR SEQUENTIAL? OR CONSECUTIVE? OR LOGICAL)(2N)(-
            INSTRUCTION? OR STATEMENT? OR PERATION? OR PROCEDURE? OR ROUT-
            INE?)
S4     383409   REPEAT? OR REDO??? OR REPETITION OR DUPLICAT?
S5      42041   (FIXED OR DEFINITE OR DETERMINATE OR LIMITED OR CERTAIN OR
            FIRM OR SET)(3N) NUMBER
S6    3371540   TIME? OR INTERVAL? OR DURATION? OR DECREMENT? OR CLOCK OR -
            SCHEDULE?
S7        946   S1 AND S2 AND S3
S8          1   S7 AND S4 AND S5 AND S6
S9          1   S3 AND S4 AND (S5 (2N) S6)
S10        97   S7 AND S4
S11         1   S10 AND S5
S12         9   S7 AND S5
S13        10   S8 OR S9 OR S11 OR S12
File 347:JAPIO Nov 1976-2004/Apr(Updated 040802)
         (c) 2004 JPO & JAPIO
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200450
         (c) 2004  Thomson Derwent
```

13/5/1 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

04934157 **Image available**
SLAVE STATION NUMBER AUTOMATIC SETTING SYSTEM

PUB. NO.: 07-226757 [JP 7226757 A]
PUBLISHED: August 22, 1995 (19950822)
INVENTOR(s): SHIBAYAMA TSUTOMU
APPLICANT(s): FUJITSU DENSO LTD [470928] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 06-019191 [JP 9419191]
FILED: February 16, 1994 (19940216)
INTL CLASS: [6] H04L-012/42
JAPIO CLASS: 44.3 (COMMUNICATION -- Telegraphy)

ABSTRACT

PURPOSE: To check the setting state with respect to the system setting automatically each of plural slave stations from a master station on **duplicated** ring transmission lines.

CONSTITUTION: In the communication system where a master station 100 and plural slave stations 200 are arranged on **duplicated** ring transmission lines 300, when an individual number is set to each slave station by remote control from the master station, the master station 100 sends unidirectionally a command incrementing or decrementing sequentially the individually of each slave station 200 and an initial value of the individual number, sends an increment command and the initial value of the individual number in the opposite direction, each slave station 200 sets each individual number respectively while incrementing or decrementing a number to/from the initial value sequentially and returns the set individual number to the master station and the master station sends a command of increment or decrement and the value incrementing or **decrementing** a **number** to/from the **set** individual **number** to a succeeding slave station 200. The **procedure** above is **sequentially** conducted.

13/5/2 (Item 2 from file: 347)
DIALOG(R)File 347:JAPIO

04800551 **Image available**
INSTRUCTION SUPPLYING DEVICE

PUB. NO.: 07-093151 [JP 7093151 A]
PUBLISHED: April 07, 1995 (19950407)
INVENTOR(s): KAWAI ATSUSHI
APPLICANT(s): OKI ELECTRIC IND CO LTD [000029] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 05-257710 [JP 93257710]
FILED: September 21, 1993 (19930921)
INTL CLASS: [6] G06F-009/38
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)

ABSTRACT

PURPOSE: To shorten program execution time while avoiding the insertion of a no- **operation** instruction as much as possible and further to make the number of no- **operation** instructions fixed without depending on pipeline structure,

CONSTITUTION: Normal **instructions** are **sequentially** executed as conventional, but when there is a no-condition branching instruction, the order of instruction execution is exchanged by an order control part 30 so as not to **generate** the fixing of a branching destination, instruction address and time waiting up to the call of a branching destination

instruction. When no branch is **generated** in the execution of the condition branching instruction, following **instructions** to be **sequentially** executed are first read out without performing the fixing of the branching destination instruction address, the fixing of branching conditions and time waiting up to the call of the branching destination instruction. When any branch is **generated** by the condition branching instruction, the **fixed number** of no- **operation** instructions are automatically inserted by a nop instruction inserting part 21.

13/5/3 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

016196670 **Image available**
WPI Acc No: 2004-354556/200433
XRPX Acc No: N04-283244
Interrupts handling apparatus for translation of Java instructions in embedded system, presents translated codes of instruction code to processor and cache with respect to limited number **of addresses**
Patent Assignee: LSI LOGIC CORP (LSIL-N)
Inventor: COHEN A; PERETS R; ZEMLYAK B
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6718539	B1	20040406	US 2000748641	A	20001222	200433	B

Priority Applications (No Type Date): US 2000748641 A 20001222
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6718539	B1		19	G06F-009/44	

Abstract (Basic): US 6718539 B1
NOVELTY - The instructions codes of Java instruction set is translated into instruction codes of specific instruction set. The translated codes are presented to processor and FIFO based cache with respect to **limited number** of addresses such that presentation of instruction codes is unaffected by interrupt to processor, and translator circuit **generates** null **operation** instruction (NOP) code after jump instruction code.
DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for interrupts handling method.
USE - For seamlessly handling interrupts during translation of Java instruction set into specific instruction set such as C/C++ instruction set, in embedded system.
ADVANTAGE - Efficiently translates unlimited program space of processor. Thereby, achieving fast execution of Java instructions and maintaining backward compatibility for legacy code.
DESCRIPTION OF DRAWING(S) - The figure shows a block diagram of the interrupt handling apparatus.
CPU (102)
memory system (104)
CPU interface (152)
memory interface (154)
registers(214) **instruction sequence** cache (200)
pp; 19 DwgNo 3/11
Title Terms: INTERRUPT; HANDLE; APPARATUS; TRANSLATION; INSTRUCTION; EMBED; SYSTEM; PRESENT; TRANSLATION; CODE; INSTRUCTION; CODE; PROCESSOR; CACHE; RESPECT; LIMIT; NUMBER; ADDRESS
Derwent Class: T01
International Patent Class (Main): G06F-009/44
International Patent Class (Additional): G06F-011/00
File Segment: EPI

13/5/4 (Item 2 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013881807 **Image available**
WPI Acc No: 2001-366019/200138
XRPX Acc No: N01-266932

Computer system, has controller that converts logical address to physical address that selectively displaces data inward or outward

Patent Assignee: MICRON TECHNOLOGY INC (MICR-N)
Inventor: KLEIN D A
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6202118	B1	20010313	US 97926797	A	19970910	200138	B

Priority Applications (No Type Date): US 97926797 A 19970910
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6202118	B1		16	G06F-012/10	

Abstract (Basic): US 6202118 B1

NOVELTY - A controller manages data transfer between a host computer and a disk storage device. The disk storage device has at least one outer track capable of a higher data transfer rate than at least one inner track.

DETAILED DESCRIPTION - The controller has an associated memory that stores program instructions which when executed cause the controller to receive a data transfer command that includes data and a logical address that is part of a sequence of logical address, to perform a logical to physical translation on the logical address to **produce** a resulting physical address that selectively displaces the data inward or outward, out of the sequence of logical addresses responsive to a determination of whether the data does or does not require the higher data transfer rate, and to transfer data between the host computer and disk storage device at the resulting physical address. INDEPENDENT CLAIMS are also included for the following:

(a) a method for storing data on a disk storage device;
(b) a disk storage device;
(c) and a controller for storing data on a disk storage device.

USE - Computer system having disk storage device with improved data storage scheme.

ADVANTAGE - Improves overall transfer rate of user data with the translation of **certain** low- **number** logical addresses into higher number physical addresses in order to displace inward the low-number logical data. Enables storage of at least some user data on higher transfer rate, outer cylinders.

DESCRIPTION OF DRAWING(S) - The figure is a flowchart depicting a **routine** for implementing **logical** to physical address translation in the computer system.

pp; 16 DwgNo 5/8

Title Terms: COMPUTER; SYSTEM; CONTROL; CONVERT; LOGIC; ADDRESS; PHYSICAL; ADDRESS; SELECT; DISPLACE; DATA; INWARD; OUTWARD
Derwent Class: T01
International Patent Class (Main): G06F-012/10
File Segment: EPI

13/5/5 (Item 3 from file: 350)
DIALOG(R)File 350:Derwent WPIX

012769373 **Image available**
WPI Acc No: 1999-575596/199949
XRPX Acc No: N99-424755

Automatic sequence **program** generating procedure **- involves setting I/O register number into control signal table provided at memory of motion controller, and reads register number to** generate **corresponding sequence program**

Patent Assignee: YASKAWA ELECTRIC CORP (YASW)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 11249715	A	19990917	JP 9845605	A	19980226	199949	B

Priority Applications (No Type Date): JP 9845605 A 19980226
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 11249715	A		8	G05B-019/05	

Abstract (Basic): JP 11249715 A
NOVELTY - An input-output I/O register **number** is **set** into a control signal table provided at the memory of a motion controller. This number is then read by an automatic sequence program **generator** which in turn **generates** the corresponding sequence program. DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for the motion controller of a sequence program.
USE - For automatically **generating** sequence program used to run motion programs.
ADVANTAGE - Automates sequence program **developing** process since user can concentrate on **developing** motion program without being conscious of **operation** of sequence mechanism of programmable controller. DESCRIPTION OF DRAWING(S) - The figure shows the process drawing of the automatic sequence program **generator** .
Dwg.1/5
Title Terms: AUTOMATIC; SEQUENCE; PROGRAM; **GENERATE** ; **PROCEDURE** ; SET; REGISTER; NUMBER; CONTROL; SIGNAL; TABLE; MEMORY; MOTION; CONTROL; READ; REGISTER; NUMBER; **GENERATE** ; CORRESPOND; SEQUENCE; PROGRAM
Derwent Class: T01; T06
International Patent Class (Main): G05B-019/05
International Patent Class (Additional): G05B-015/02; G05B-019/02; G05B-019/18
File Segment: EPI

13/5/6 (Item 4 from file: 350)
DIALOG(R)File 350:Derwent WPIX

010691637 **Image available**
WPI Acc No: 1996-188593/199619
XRPX Acc No: N96-157690
Integrated circuit data processing appts with interface circuit - has interface to allow external appts to signal request for communication with processor which communicates with requests during predetermined periods during execution of stored program
Patent Assignee: CAMBRIDGE CONSULTANTS (CAMB-N); CAMBRIDGE SILICON RADIO LTD (CAMB-N)
Inventor: BARLOW S J; COLLIER J D Y; MORFEY A G; COLLIER J; COLLIER J D
Number of Countries: 066 Number of Patents: 013
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 9609583	A2	19960328	WO 95GB2283	A	19950925	199619	B
GB 2294137	A	19960417	GB 9519543	A	19950925	199619	
GB 2294138	A	19960417	GB 9519544	A	19950925	199619	
AU 9535305	A	19960409	AU 9535305	A	19950925	199629	
WO 9609583	A3	19960606	WO 95GB2283	A	19950925	199633	
GB 2294137	B	19970115	GB 9519543	A	19950925	199706	
GB 2294138	B	19970115	GB 9519544	A	19950925	199706	
EP 787321	A1	19970806	EP 95932126	A	19950925	199736	
			WO 95GB2283	A	19950925		
US 6311263	B1	20011030	WO 95GB2283	A	19950925	200172	
			US 97809498	A	19970324		
EP 1189144	A2	20020320	EP 95932126	A	19950925	200227	
			EP 2001204179	A	19950925		
EP 787321	B1	20020626	EP 95932126	A	19950925	200242	

			WO 95GB2283	A	19950925		
			EP 2001204179	A	19950925		
DE 69527210	E	20020801	DE 627210	A	19950925	200258	
			EP 95932126	A	19950925		
			WO 95GB2283	A	19950925		
US 20020161988	A1	20021031	US 97809498	A	19970324	200274	
			US 200132985	A	20011029		

Priority Applications (No Type Date): GB 9419246 A 19940923
Cited Patents: 2.Jnl.Ref; EP 55370; US 4342082; US 5274770
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 9609583	A2	E	86	G06F-009/30	
Designated States (National): AM AT AU BB BG BR BY CA CH CN CZ DE DK EE ES FI GB GE HU IS JP KE KG KP KR KZ LK LR LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK TJ TM TT UA UG US UZ VN					
Designated States (Regional): AT BE CH DE DK ES FR GB GR IE IT KE LU MC MW NL OA PT SD SE SZ UG					
GB 2294137	A		88	G06F-013/14	
GB 2294138	A		77	G06F-007/48	
AU 9535305	A			G06F-009/30	Based on patent WO 9609583
WO 9609583	A3			G06F-009/30	
GB 2294137	B			G06F-013/14	
GB 2294138	B		1	G06F-007/48	
EP 787321	A1	E		G06F-009/30	Based on patent WO 9609583
Designated States (Regional): AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE					
US 6311263	B1			G06F-013/364	Based on patent WO 9609583
EP 1189144	A2	E		G06F-015/78	Div ex application EP 95932126 Div ex patent EP 787321
Designated States (Regional): AT BE CH DE DK FR GB GR IE IT LI LU MC NL PT SE					
EP 787321	B1	E		G06F-009/30	Related to application EP 2001204179 Related to patent EP 1189144 Based on patent WO 9609583
Designated States (Regional): AT BE CH DE DK FR GB IT LI NL SE					
DE 69527210	E			G06F-009/30	Based on patent EP 787321 Based on patent WO 9609583
US 20020161988	A1			G06F-009/00	Cont of application US 97809498 Cont of patent US 6311263

Abstract (Basic): WO 9609583 A

The integrated circuit includes a microprocessor core, program memory (104) and separate data memory (106,108) and analogue and digital signal processing circuit (110). The ALU is 16 bits wide, and a 32 bit shift register is provided using a pair of 16 bit registers. The processor has a fixed length instruction format, with an instruction set including multiply and divide **operations** which use the shift register over several cycles.

External pins of the IC allow for single stepping and debug **operations** , and a serial interface (SIF) allows external communication of test data or working data as necessary. The serial interface has four wires (SERIN, SEROUT, SER-CLK, SERLOADB), allowing handshaking with a master appts, and allowing direct access to the memory space (104-110) of the processor core, without specific program control. Within each processor cycle, the processing circuit is divided into several stages, and latches are interposed between the stages.

USE/ADVANTAGE - Microprocessor architecture for integration within ASIC. Allows verifiable design and programming such that risk of errors in ASIC design are minimised.

Dwg.7/12

Title Terms: INTEGRATE; CIRCUIT; DATA; PROCESS; APPARATUS; INTERFACE; CIRCUIT; INTERFACE; ALLOW; EXTERNAL; APPARATUS; SIGNAL; REQUEST; COMMUNICATE; PROCESSOR; COMMUNICATE; REQUEST; PREDETERMINED; PERIOD; EXECUTE; STORAGE; PROGRAM
Derwent Class: T01
International Patent Class (Main): G06F-007/48; G06F-009/00; G06F-009/30;

G06F-013/14; G06F-013/364; G06F-015/78
International Patent Class (Additional): G01F-001/66; G06F-005/00; G06F-009/302; G06F-009/44; G06F-009/455; G06F-012/02; G06F-013/12; G06F-013/16; G06F-013/36; G06F-013/38; G06F-013/42
File Segment: EPI

13/5/7 (Item 5 from file: 350)
DIALOG(R)File 350:Derwent WPIX

010353430 **Image available**
WPI Acc No: 1995-254744/199533
XRPX Acc No: N95-196700
Channel appts in data processing system - has request data generation **section responsive to transfer instruction from controller and input ready signal**
Patent Assignee: TOSHIBA KK (TOKE)
Inventor: KIHARA J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 5432912	A	19950711	US 89399939	A	19890829	199533	B
			US 92921603	A	19920803		
			US 9377958	A	19930618		

Priority Applications (No Type Date): JP 88216812 A 19880831
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5432912	A		18	G06F-013/00	Cont of application US 89399939
					Cont of application US 92921603

Abstract (Basic): US 5432912 A
The appts includes a request information **generating** device responsive to a transfer **instruction** for **consecutively generating** a **set number** of read request information units. Each of the read request information units contains a memory address from the updating device and a source identifier. The latter includes a device identifier and a within-device identifier. The device identifier identifies an alternate storage device, and the within-device identifier identifies a relative order in which the read request information unit was **generated** .
A receiving device sequentially receives a number of response information units output from the main storage device in response to the read request information units.
USE/ADVANTAGE - For rearranging received read data in order of **generation** of addresses which are used in split bus control system. Speeds up memory read-out **operation** .
Dwg.2/5
Title Terms: CHANNEL; APPARATUS; DATA; PROCESS; SYSTEM; REQUEST; DATA; **GENERATE** ; SECTION; RESPOND; TRANSFER; INSTRUCTION; CONTROL; INPUT; READY ; SIGNAL
Derwent Class: T01
International Patent Class (Main): G06F-013/00
File Segment: EPI

13/5/8 (Item 6 from file: 350)
DIALOG(R)File 350:Derwent WPIX

008469380 **Image available**
WPI Acc No: 1990-356380/199048
Related WPI Acc No: 1999-169372
XRPX Acc No: N90-272192
Multiple instruction issue computer architecture - has series of pipeline stages and includes resources for accepting family of instructions at each stage

Patent Assignee: TANDEM COMPUTERS INC (TAND); TANDEM COMPUTERS IN (TAND-N); HORST R W (HORS-I)
Inventor: HORST R W; HORSE R W; JARDINE R L; LYNCH S J; MANELA P R
Number of Countries: 008 Number of Patents: 015
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 399762	A	19901128	EP 90305489	A	19900521	199048	B
AU 9055153	A	19901129				199104	
CA 2016068	A	19901124				199107	
EP 399762	A3	19920122				199322	
US 5390355	A	19950214	US 89356170	A	19890524	199512	
			US 92890299	A	19920527		
US 5574941	A	19961112	US 89356170	A	19890524	199651	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95552433	A	19951103		
US 5628024	A	19970506	US 89356170	A	19890524	199724	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
US 5752064	A	19980512	US 89356170	A	19890524	199826	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
			US 96710620	A	19960920		
US 5918032	A	19990629	US 89356170	A	19890524	199932	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
			US 96710620	A	19960920		
			US 97959643	A	19971028		
EP 399762	B1	19991222	EP 90305489	A	19900521	200004	
			EP 98122872	A	19900521		
US 6009506	A	19991228	US 89356170	A	19890524	200007	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
			US 96710620	A	19960920		
			US 97959643	A	19971028		
			US 9859271	A	19980410		
DE 69033398	E	20000127	DE 633398	A	19900521	200012	
			EP 90305489	A	19900521		
CA 2016068	C	20000404	CA 2016068	A	19900504	200035	
US 6092177	A	20000718	US 89356170	A	19890524	200037	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
			US 96710620	A	19960920		
			US 97959643	A	19971028		
			US 99257883	A	19990225		
US 6266765	B1	20010724	US 89356170	A	19890524	200146	
			US 92890299	A	19920527		
			US 94300815	A	19940902		
			US 95483661	A	19950607		
			US 96710620	A	19960920		
			US 97959643	A	19971028		
			US 99257883	A	19990225		
			US 2000611378	A	20000707		

Priority Applications (No Type Date): US 89356170 A 19890524; US 92890299 A 19920527; US 94300815 A 19940902; US 95552433 A 19951103; US 95483661 A 19950607; US 96710620 A 19960920; US 97959643 A 19971028; US 9859271 A 19980410; US 99257883 A 19990225; US 2000611378 A 20000707
Cited Patents: NoSR.Pub; 2.Jnl.Ref; EP 118830; EP 155211; EP 239081; EP 354740; EP 71028
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes

EP 399762	A	34		
Designated States (Regional): DE FR GB IT SE				
EP 399762	A3	34		
US 5390355	A	30	G06F-009/38	Cont of application US 89356170
US 5574941	A	30	G06F-009/30	Cont of application US 89356170
				Cont of application US 92890299
				Cont of application US 94300815
				Cont of patent US 5390355
US 5628024	A	31	G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of patent US 5390355
US 5752064	A		G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of application US 95483661
				Cont of patent US 5390355
				Cont of patent US 5628024
US 5918032	A		G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of application US 95483661
				Cont of application US 96710620
				Cont of patent US 5390355
				Cont of patent US 5628024
				Cont of patent US 5752064
EP 399762	B1 E		G06F-009/38	Related to application EP 98122872
				Related to patent EP 902362
Designated States (Regional): DE FR GB IT SE				
US 6009506	A		G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of application US 95483661
				Cont of application US 96710620
				Div ex application US 97959643
				Cont of patent US 5390355
				Div ex patent US 5574941
				Cont of patent US 5628024
				Cont of patent US 5752064
DE 69033398	E		G06F-009/38	Based on patent EP 399762
CA 2016068	C E		G06F-009/28	
US 6092177	A		G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of application US 95483661
				Cont of application US 96710620
				Cont of application US 97959643
				Cont of patent US 5390355
				Cont of patent US 5628024
				Cont of patent US 5752064
				Cont of patent US 5918032
US 6266765	B1		G06F-009/38	Cont of application US 89356170
				Cont of application US 92890299
				Div ex application US 94300815
				Cont of application US 95483661
				Cont of application US 96710620
				Cont of application US 97959643
				Cont of application US 99257883
				Cont of patent US 5390355
				Cont of patent US 5628024
				Cont of patent US 5752064
				Cont of patent US 5918032
				Cont of patent US 6092177

Abstract (Basic): EP 399762 A

In a data processor, the improved instruction processing system facilitates processing instructions at a rate of more than one

instruction per clock. The system comprises a device for fetching a familty of n, n being a predetermined integer, **sequential instructions** in a program. A pipeline, having a series of pipeline stages, includes resources for accepting a family of n instructions at each pipeline stage during a single clock.

The accepted family of the type includes multiple nonbranching instructions, with one of the nonbranching instructions being a memory reference type of instructions, so that the pipeline can retire a family of instructions and accept a new family of instructions.

ADVANTAGE - Increased instruction processing throughput.

Dwg.1/14

Title Terms: MULTIPLE; INSTRUCTION; ISSUE; COMPUTER; ARCHITECTURE; SERIES; PIPE; STAGE; RESOURCE; ACCEPT; FAMILY; INSTRUCTION; STAGE
Derwent Class: T01
International Patent Class (Main): G06F-009/28; G06F-009/30; G06F-009/38
International Patent Class (Additional): G06F-009/22
File Segment: EPI

13/5/9 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX

007326169
WPI Acc No: 1987-323176/198746
XRPX Acc No: N87-241695

Stack frame cache on microprocessor chip - minimises main memory references initiated during execution of call-return instructions

Patent Assignee: INTEL CORP (ITLC)
Inventor: HINTON G; IMEL M T; LAI K; MEYERS G J; RICHES R; MYERS G J
Number of Countries: 005 Number of Patents: 007
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
GB 2190521	A	19871118	GB 8628175	A	19861125	198746	B
DE 3716229	A	19871119	DE 3716229	A	19870514	198747	
FR 2598835	A	19871120				198803	
CN 8700507	A	19871125				198847	
US 4811208	A	19890307	US 86863878	A	19860516	198912	
GB 2190521	B	19900110				199002	
DE 3716229	C2	19960814	DE 3716229	A	19870514	199637	

Priority Applications (No Type Date): US 86863878 A 19860516
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 4811208	A		9		
DE 3716229	C2		11	G06F-009/34	

Abstract (Basic): GB 2190521 A

Global registers (21) are provided on the microprocessor chip. One of them is a frame pointer register containing the current frame pointer, and the remainder are available to a current process as general registers. Floating point registers are also provided for use by the current process in execution of floating point arithmetic **operations** by a processor (22). A register set pool (23), forming an on-chip cache and made-up of register sets is provided, each register **set** comprising a **number** of local registers.

When a call instruction is decoded, a register set of local registers from the register set pool is allocated to the called **procedure** , and the frame pointer register is initialised. When a return instruction is decoded, the register set is freed for allocation to another **procedure** called by a subsequent call instruction.

Title Terms: STACK; FRAME; CACHE; MICROPROCESSOR; CHIP; MINIMISE; MAIN; MEMORY; REFERENCE; INITIATE; EXECUTE; CALL; RETURN; INSTRUCTION
Derwent Class: T01
International Patent Class (Main): G06F-009/34
International Patent Class (Additional): G06F-009/30; G06F-009/40; G06F-012/08; G06F-015/06

File Segment: EPI

13/5/10 (Item 8 from file: 350)
DIALOG(R)File 350:Derwent WPIX

007317911
WPI Acc No: 1987-314918/198745
XRPX Acc No: N87-235696

Performance optimisation in fixed clock rate computer system - executing instruction in control word simultaneously with counting value in control word for signalling end of decrementing

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC); IBM CORP (IBMC)
Inventor: NGAI C H; WATKINS G J
Number of Countries: 005 Number of Patents: 004
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 244676	A	19871111	EP 87105554	A	19870414	198745	B
US 4868739	A	19890919	US 86859557	A	19860505	198947	
EP 244676	B1	19940914	EP 87105554	A	19870414	199435	
DE 3750520	G	19941020	DE 3750520	A	19870414	199441	
			EP 87105554	A	19870414		

Priority Applications (No Type Date): US 86859557 A 19860505
Cited Patents: A3...9149; EP 135721; GB 2162406; No-SR.Pub; US 4439829
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 244676	A	E	12		
Designated States (Regional): DE FR GB IT					
US 4868739	A		11		
EP 244676	B1	E	13	G06F-009/22	
Designated States (Regional): DE FR GB IT					
DE 3750520	G			G06F-009/22	Based on patent EP 244676

Abstract (Basic): EP 244676 A

The control word (30) is of two parts of which the first (54) consists of **operational** instructions and the second (56) provides control data and timing data. A counter is connected with a price control store and operates to count the number of cycles (48) in the control word. As the control word is being executed, the instructions (54) are passed to element processors via a command bus and register.

The counter is loaded with the number of cycles (48) in the control word and decrements the count as the corresponding control word is executed. At the time when the counter decrements to zero it provides an end-of-decrementing **operation** signal to the price control store. The store is then in a position to start execution of the next following control word.

ADVANTAGE - Reduces time between **operations** .

Title Terms: PERFORMANCE; OPTIMUM; FIX; CLOCK; RATE; COMPUTER; SYSTEM; EXECUTE; INSTRUCTION; CONTROL; WORD; SIMULTANEOUS; COUNT; VALUE; CONTROL; WORD; SIGNAL; END
Derwent Class: T01
International Patent Class (Main): G06F-009/22
File Segment: EPI

```
Set     Items   Description
S1    1424624   CREAT? OR GENERAT? OR PRODUC? OR DEVELOP?
S2    1079600   MACRO()INSTRUCTION? OR STATEMENT? OR OPERATION? OR PROCEDU-
            RE? OR ROUTINE? OR MACRO
S3      18176   (SEQUENCE? OR SEQUENTIAL? OR CONSECUTIVE? OR LOGICAL)(2N)(-
            INSTRUCTION? OR STATEMENT? OR PERATION? OR PROCEDURE? OR ROUT-
            INE?)
S4     454994   REPEAT? OR REDO??? OR REPETITION OR DUPLICAT?
S5      85885   (FIXED OR DEFINITE OR DETERMINATE OR LIMITED OR CERTAIN OR
            FIRM OR SET)(3N) NUMBER
S6    1149469   TIME? OR INTERVAL? OR DURATION? OR DECREMENT? OR CLOCK OR -
            SCHEDULE?
S7       3392   S1 (S) (S2 (2N) S3)
S8          5   S7 (S) S4 (S) (S5 (2N) S6)
File 348:EUROPEAN PATENTS 1978-2004/Aug W01
         (c) 2004 European Patent Office
File 349:PCT FULLTEXT 1979-2002/UB=20040805,UT=20040729
         (c) 2004 WIPO/Univentio
```

```
Set     Items   Description
S1    7253670   CREAT? OR GENERAT? OR PRODUC? OR DEVELOP?
S2    2303524   MACRO()INSTRUCTION? OR STATEMENT? OR OPERATION? OR PROCEDU-
               RE? OR ROUTINE? OR MACRO
S3       5900   (SEQUENCE? OR SEQUENTIAL? OR CONSECUTIVE? OR LOGICAL)(2N)(-
               INSTRUCTION? OR STATEMENT? OR PERATION? OR PROCEDURE? OR ROUT-
               INE?)
S4     267798   REPEAT? OR REDO??? OR REPETITION OR DUPLICAT?
S5      31608   (FIXED OR DEFINITE OR DETERMINATE OR LIMITED OR CERTAIN OR
               FIRM OR SET)(3N) NUMBER
S6    3499591   TIME? OR INTERVAL? OR DURATION? OR DECREMENT? OR CLOCK OR -
               SCHEDULE?
S7       1910   S1 AND S2 AND S3
S8          0   S3 AND S4 AND (S5 (2N) S6)
S9          0   S7 AND S8
S10         1   S3 AND S4 AND S5
S11        49   S7 AND S4
S12         0   S11 AND S5
S13        18   S7 AND S5
S14        68   S10 OR S11 OR S13
S15        54   S14 NOT PY>2001
S16        54   S15 NOT PD>20010824
S17        43   RD (unique items)
File   8:Ei Compendex(R) 1970-2004/Aug W1
         (c) 2004 Elsevier Eng.  Info. Inc.
File 202:Info. Sci. & Tech. Abs. 1966-2004/Jul 12
         (c) 2004 EBSCO Publishing
File  65:Inside Conferences 1993-2004/Aug W2
         (c) 2004 BLDSC all rts. reserv.
File   2:INSPEC 1969-2004/Aug W1
         (c) 2004 Institution of Electrical Engineers
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep
         (c) 2003 EBSCO Pub.
File  94:JICST-EPlus 1985-2004/Jul W3
         (c)2004 Japan Science and Tech Corp(JST)
File  99:Wilson Appl. Sci & Tech Abs 1983-2004/Jul
         (c) 2004 The HW Wilson Co.
File  95:TEME-Technology & Management 1989-2004/Jun W1
         (c) 2004 FIZ TECHNIK
```

721 (Computer Circuits & Logic Elements); 714 (Electronic Components); 703 (Electric Circuits); 921 (Applied Mathematics)
72 (COMPUTERS & DATA PROCESSING); 71 (ELECTRONICS & COMMUNICATIONS); 70 (ELECTRICAL ENGINEERING); 92 (ENGINEERING MATHEMATICS)

17/5/5 (Item 5 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05619521 E.I. No: EIP00085275153
Title: Using path induction to evaluate sequential **allocation** procedures
Author: Hardwick, Janis P.; Stout, Quentin F.
Corporate Source: Univ of Michigan, Ann Arbor, MI, USA
Source: SIAM Journal of Scientific Computing v 21 n 1 1999. p 67-87
Publication Year: 1999
CODEN: SJOCE3 ISSN: 1064-8275
Language: English
Document Type: JA; (Journal Article) Treatment: A; (Applications); T; (Theoretical)
Journal Announcement: 0009W3
Abstract: Path induction is a technique used to speed the process of making multiple exact evaluations of a **sequential** allocation **procedure** , where the options are discrete and their outcomes follow a discrete distribution. Multiple evaluations are needed for determining criteria such as maxima or minima over parameter regions (where the location of the extremal value is unknown in advance), for visualizing characteristics such as robustness, or for obtaining the distribution of a statistic rather than just its mean. By using an initial phase to determine the number of paths reaching each terminal state, the subsequent evaluations are far faster than **repeated** use of standard evaluation techniques. Algorithms are given for fully sequential and staged **sequential procedures** , and the **procedures** can be either deterministic or random. The **procedures** can be **generated** by any technique (including dynamic programming or ad hoc approaches), and the evaluations performed can be quite flexible and need not be related to the method of obtaining the **procedure** . While the emphasis is on path induction, the techniques used to speed up the analyses of staged allocation **procedures** can also be used to improve backward induction for such **procedures** . If multiple evaluations need to be carried out, however, path induction will still be far superior. For each parameter configuration to be evaluated, one reduces the time by a factor of n, where n is the size of the experiment, by using path induction rather than the standard technique of backward induction. In some settings the savings is significantly greater than n. (Author abstract) 17 Refs.
Descriptors: *Dynamic programming; Random processes; Critical path analysis; PERT; Probability distributions; Algorithms; Frequency domain analysis; Sampling
Identifiers: Path induction; Sequential allocation; Bayesian bandit problems; Group sampling
Classification Codes:
921.5 (Optimization Techniques); 922.1 (Probability Theory); 723.2 (Data Processing); 912.3 (Operations Research); 921.6 (Numerical Methods); 922.2 (Mathematical Statistics)
921 (Applied Mathematics); 922 (Statistical Methods); 723 (Computer Software); 912 (Industrial Engineering & Management)
92 (ENGINEERING MATHEMATICS); 72 (COMPUTERS & DATA PROCESSING); 91 (ENGINEERING MANAGEMENT)

17/5/8 (Item 8 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05063586 E.I. No: EIP98074289946
Title: Exact calculations for the repeated **many-one test**
Author: Wang, Qin

Corporate Source: Univ of Cincinnati Medical Cent, Cincinnati, OH, USA
Source: Computational Statistics & Data Analysis v 27 n 4 Jun 5 1998. p 389-399
Publication Year: 1998
CODEN: CSDADW ISSN: 0167-9473
Language: English
Document Type: JA; (Journal Article) Treatment: T; (Theoretical)
Journal Announcement: 9809W2
Abstract: When group sequential methods are applied to compare several treatments with a control, multiplicity from the **repeated** significance testing as well as from the multiple comparison has to be accounted for properly to control the overall type I error. This often involves a multidimensional integration **procedure** to compute group sequential boundaries. Liu proposed some group **sequential procedures** for comparing several treatments with a control. The group sequential boundaries needed to apply these **procedures** were estimated by simulation to two decimal places for specified alpha -levels. In this paper, we describe an algorithm that allows efficient calculation of these boundaries by combining a newly **developed** software in Genz and Keister and the dynamic programming technique described, for example, in Cormen et al. The algorithm is then implemented to successfully solve the computation problem which is previously stated by Liu (1996) as `no simple exact calculation method is available'. (Author abstract) 16 Refs.
Descriptors: *Control theory; Computational methods; Integration; Computer simulation; Algorithms; Computer software; Dynamic programming; Problem solving
Identifiers: Group sequential methods
Classification Codes:
731.1 (Control Systems); 921.2 (Calculus); 723.5 (Computer Applications); 921.5 (Optimization Techniques)
731 (Automatic Control Principles); 921 (Applied Mathematics); 723 (Computer Software)
73 (CONTROL ENGINEERING); 92 (ENGINEERING MATHEMATICS); 72 (COMPUTERS & DATA PROCESSING)

17/5/11 (Item 11 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04272306 E.I. No: EIP95092858836
Title: Projection pursuit for high dimensional feature reduction: parallel and sequential approaches
Author: Jimenez, Luis O.; Landgrebe, David A.
Corporate Source: Purdue Univ, West Lafayette, IN, USA
Conference Title: Proceedings of the 1995 International Geoscience and Remote Sensing Symposium. Part 1 (of 3)
Conference Location: Firenze, Italy Conference Date: 19950710-19950714
Sponsor: IEEE; URSI
E.I. Conference No.: 43564
Source: International Geoscience and Remote Sensing Symposium (IGARSS) v 1 1995.,95CH35770. p 148-150
Publication Year: 1995
CODEN: IGRSE3
Language: English
Document Type: CA; (Conference Article) Treatment: A; (Applications); T ; (Theoretical)
Journal Announcement: 9512W3
Abstract: Supervised classification techniques use labeled samples in order to train the classifier. Usually the number of such samples is **limited** , and as the **number** of bands available increases, this limitation becomes more severe, and can become dominate over the projected added value of having the additional bands available. This suggests the need for reducing the dimensionality via a preprocessing method. Such reduction should enable the estimation of feature extraction parameters to be more accurate. Using a technique referred to as Projection Pursuit, two parametric projection pursuit algorithms have been **developed** : Parallel

Parametric Projection Pursuit and Sequential Parametric Projection Pursuit. In the present paper both methods are presented, and an iterative **procedure** of the **Sequential** Approach that mitigates the computation time problem is shown. (Author abstract) 6 Refs.

Descriptors: *Algorithms; Remote sensing; Feature extraction; Image analysis; Estimation; Computational methods; Matrix algebra; Constraint theory; Optimization

Identifiers: Projection pursuit; Sequential parametric projection; Projection index; Maximum Bhattacharyya distance

Classification Codes:

921.6 (Numerical Methods); 732.2 (Control Instrumentation); 723.2 (Data Processing); 921.1 (Algebra); 721.1 (Computer Theory, Includes Formal Logic, Automata Theory, Switching Theory, Programming Theory); 921.5 (Optimization Techniques)

921 (Applied Mathematics); 732 (Control Devices); 723 (Computer Software); 721 (Computer Circuits & Logic Elements)

92 (ENGINEERING MATHEMATICS); 73 (CONTROL ENGINEERING); 72 (COMPUTERS & DATA PROCESSING)

17/5/12 (Item 12 from file: 8)

DIALOG(R)File 8:Ei Compendex(R)

02510177 E.I. Monthly No: EI8802015132

Title: BUILDING A RANDOM-NUMBER GENERATOR .

Author: Wichmann, Brian; Hill, David

Corporate Source: NPL, Teddington, Engl

Source: Byte v 12 n 3 Mar 1987 p 127-128

Publication Year: 1987

CODEN: BYTEDJ ISSN: 0360-5280

Language: ENGLISH

Document Type: JA; (Journal Article) Treatment: T; (Theoretical)

Journal Announcement: 8802

Abstract: The random-number sequences that computers **produce** are not truly random at all since true randomness depends upon having a random process available, such as tossing a coin. Because such a process is not available within a computer, a pseudorandom process is used to **produce** sequences of numbers that appear to be random and mimic well the behavior that is expected of true random **sequences** . A Pascal **routine** for very-long-cycle random-number sequences and the **repeatability** and efficency of the random-number **generator** is considered. The algorithm is written in a high-level language so that the code is portable from one machine to another without difficulty. The algorithm presented here meets all these criteria. 3 refs.

Descriptors: MATHEMATICAL STATISTICS--*Random Number **Generation** ; COMPUTER METATHEORY--Probabilistic Logics; COMPUTER PROGRAMMING--Algorithms ; COMPUTER SOFTWARE--Portability; COMPUTER PROGRAMMING LANGUAGES--PASCAL

Identifiers: RANDOM-NUMBER **GENERATOR** ; RANDOM-NUMBER SEQUENCES; PSEUDORANDOM PROCESS

Classification Codes:

922 (Statistical Methods); 723 (Computer Software)

92 (ENGINEERING MATHEMATICS); 72 (COMPUTERS & DATA PROCESSING)

17/5/13 (Item 13 from file: 8)

DIALOG(R)File 8:Ei Compendex(R)

02293556 E.I. Monthly No: EI8707066950

Title: IMPLEMENTING THE KERNEL FOR THE CONCURRENT DISTRIBUTED LANGUAGE MML ON DIFFERENT MICROPROCESSORS.

Author: Ferrari, R.; Tagliabue, M.; Zoccolante, L.; Reghizzi, S. Crespi

Corporate Source: Politecnico di Milano, Milan, Italy

Source: Microprocessing and Microprogramming v 20 n 1-3 Apr 1987, Short Notes, Euromicro '86, Venice, Italy, 1986 p 91-97

Publication Year: 1987

CODEN: MMICDT
Language: ENGLISH
Document Type: JA; (Journal Article) Treatment: A; (Applications); T; (Theoretical)
Journal Announcement: 8707
Abstract: The MML project (Multi Micro programming Line) has realized a portable Multi Micro **Development** System (MMDS) for **developing** concurrent distributed programs for multiple microprocessor architecture. This system includes a Pascal-like modular language extended to express processes and their interactions. An MML program consists of a **fixed number** of parallel activities, called sequences, which start simultaneously, exist forever and cooperate. Each sequence consists of static data and some quasi atomic **operations** on these. These **operations** are said controlled **procedures** , start **procedures** (at most one for each **sequence**) or interrupt **procedures** . 5 refs.
Descriptors: *COMPUTER PROGRAMMING LANGUAGES--*Performance; COMPUTER ARCHITECTURE
Identifiers: KERNEL; DISTRIBUTED LANGUAGE; MML PROJECT; DISTRIBUTED PROGRAMS; MICROPROCESSOR ARCHITECTURE
Classification Codes:
723 (Computer Software); 722 (Computer Hardware)
72 (COMPUTERS & DATA PROCESSING)

17/5/17 (Item 17 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

01175871 E.I. Monthly No: EI8212105546 E.I. Yearly No: EI82018640
Title: OPTIMAL CODE FROM FLOW GRAPHS.
Author: Ramanath, M. V. S.; Solomon, Marvin
Corporate Source: Univ of West Ont, London, Can
Source: Computer Languages v 7 n 1 1982 p 41-52
Publication Year: 1982
CODEN: COLADA ISSN: 0096-0551
Language: ENGLISH
Journal Announcement: 8212
Abstract: This study considers the problem of **generating** a linear **sequence** of **instructions** from a flow graph so as to minimize the number of jumps. The authors show that for programs constructed from atomic **statements** with semicolon, if-then, if-then-else, and **repeat** -until, the minimal number of unconditional jumps is bounded from above by e PLUS 1, where e is the number of if-then-else **statements** and b is the number of **repeat** -until **statements** . The authors show that these bounds are tight and present a linear-time algorithm for finding the optimal translation of such a program. 9 refs.
Descriptors: *COMPUTER PROGRAMMING; CODES, SYMBOLIC
Classification Codes:
723 (Computer Software)
72 (COMPUTERS & DATA PROCESSING)

17/5/18 (Item 18 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

01087127 E.I. Monthly No: EI8201001488 E.I. Yearly No: EI82019446
Title: PERFORMANCE ANALYSIS OF A NEW CLASS OF INTERLEAVED MEMORY SYSTEMS.
Author: Ghozati, S. A.
Corporate Source: Queens Coll, Flushing, NY, USA
Source: Conf Rec Asilomar Conf Circuits Syst Comput 14th, Pacific Grove, Calif, Nov 17-19 1980. Publ by IEEE Comput Soc Press (Cat n 80CH1625-3), Los Alamitos, Calif, 1981 p 459-461
Publication Year: 1980
CODEN: RACSDI ISSN: 0736-5861
Language: ENGLISH
Journal Announcement: 8201

Abstract: A new design is presented for an interleaved memory system which improves substantially the bandwidth of the memory-processor linkage over standard systems. In this new memory system a **sequence** of **instruction** and data requests issued by a set of processors is stored in a queue and a scanner examines and transfers them to requested memory modules (MM). A second set of MM exist which may **duplicate** the contents of those MM which are requested more frequently. Thus the information in the set of MM may change during each memory cycle time. Performance (bandwidth) of the system is measured by determining the average number of serviced requests the queue per memory cycle. For a **fixed number** of MMs, the bandwidth can be expressed by a function of two variables which represents the number of MM in each group. The optimal solution can be found by choosing a proper cost function. 13 refs.

Descriptors: *COMPUTER SYSTEMS, DIGITAL

Identifiers: COMPUTER MEMORIES; INTERLEAVED MEMORIES

Classification Codes:

723 (Computer Software); 722 (Computer Hardware)

72 (COMPUTERS & DATA PROCESSING)

17/5/19 (Item 19 from file: 8)

DIALOG(R)File 8:Ei Compendex(R)

00914126 E.I. Monthly No: EI8004030671 E.I. Yearly No: EI80050592

Title: INVESTIGATING THE RELIABILITY AND PRODUCTIVITY **INDICES OF AUTOMATIC LINES.**

Author: Nemirovskii, P. Z.; Bromberg, M. A.

Source: Machines & Tooling (English translation of Stanki i Instrument) v 49 n 12 1978 p 6-9

Publication Year: 1978

CODEN: MCTOAD ISSN: 0024-922X

Language: ENGLISH

Journal Announcement: 8004

Abstract: A methodology for checking the reliability and **productivity** of automatic lines with rigid links in acceptance-delivery tests is described. Measurements follow a combined plan that includes sequential tests and tests with a **fixed number** of failures. Such combination provides for maximum possible duration of testing compared with terminated **sequential procedure** . Formulae for calculating the testing plan are presented and recommendations are given for selecting plan parameters, including planning examples. 8 refs.

Descriptors: MACHINE SHOPS--* **Production** Control

Classification Codes:

604 (Metal Cutting & Machining)

60 (MECHANICAL ENGINEERING)

17/5/25 (Item 3 from file: 2)

DIALOG(R)File 2:INSPEC

5968006 INSPEC Abstract Number: C9808-4160-006

Title: Exact calculations for the repeated **many-one test**

Author(s): Qin Wang

Author Affiliation: Dept. of Epidemiology & Biostat., Cincinnati Univ. Med. Center, OH, USA

Journal: Computational Statistics & Data Analysis vol.27, no.4 p. 389-99

Publisher: Elsevier,

Publication Date: 5 June 1998 Country of Publication: Netherlands

CODEN: CSDADW ISSN: 0167-9473

SICI: 0167-9473(19980605)27:4L.389:ECRM;1-#

Material Identity Number: D994-98007

U.S. Copyright Clearance Center Code: 0167-9473/98/$19.00

Document Number: S0167-9473(98)00020-6

Language: English Document Type: Journal Paper (JP)

Treatment: Theoretical (T)
Abstract: When group sequential methods are applied to compare several treatments with a control, multiplicity from the **repeated** significance testing as well as from the multiple comparison has to be accounted for properly to control the overall type I error. This often involves a multidimensional integration **procedure** to compute group sequential boundaries. W. Liu (1996) proposed some group **sequential procedures** for comparing several treatments with a control. The group sequential boundaries needed to apply these **procedures** were estimated by simulation to two decimal places for specified alpha levels. We describe an algorithm that allows efficient calculation of these boundaries by combining a newly **developed** software by A. Genz and B.D. Keister (1996) and the dynamic programming technique described, for example, by T.H. Cormen et al. (1990). The algorithm is then implemented to successfully solve the computation problem which was previously stated by W. Liu (1996) as "no simple exact calculation method is available". (16 Refs)
Subfile: C
Descriptors: dynamic programming; integration; mathematics computing
Identifiers: exact calculations; **repeated** many-one test; group sequential methods; **repeated** significance testing; multiple comparison; type I error; multidimensional integration **procedure** ; group sequential boundaries; group **sequential procedures** ; simulation; specified alpha levels; newly **developed** software; dynamic programming technique; computation problem
Class Codes: C4160 (Numerical integration and differentiation); C7310 (Mathematics computing); C1180 (Optimisation techniques)

17/5/26 (Item 4 from file: 2)
DIALOG(R)File 2:INSPEC

03321718 INSPEC Abstract Number: C89020582
Title: Go to the root of the problem (software development)
Author(s): Allinger, W.
Journal: Elektronik vol.37, no.25 p.92-6
Publication Date: 9 Dec. 1988 Country of Publication: West Germany
CODEN: EKRKAR ISSN: 0013-5658
Language: German Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: The author reports on the use of pseudo-code as a way to better documented programs. The pseudo-code approach to the design and documentation of software gives better assembler programs and allows a cheap entry into the world of computer-aided software **development** . He lists what is required from design and documentation and then considers the following aspects of the pseudo-code solution: **statements** , **sequences** , **repetition** , selection, subroutine execution, unconditional skip and GO TO. (3 Refs)
Subfile: C
Descriptors: software engineering
Identifiers: pseudo-code technique; pseudo-code; computer-aided software **development** ; **statements** ; sequences; **repetition** ; selection; subroutine execution; unconditional skip
Class Codes: C6110B (Software engineering techniques)

17/5/27 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

02831462 INSPEC Abstract Number: C87017239
Title: Programming from first principles
Author(s): Bornat, R.
Publisher: Prentice-Hall, Englewood Cliffs, NJ, USA
Publication Date: 1987 Country of Publication: USA xviii+538 pp.
ISBN: 0 13 729104 3

Language: English Document Type: Book (BK)
Treatment: Practical (P); Theoretical (T)
Abstract: The book is a language-independent introduction to computer programming. By concentrating on analysis and proof of programs the author introduces the beginner to the sort of reasoning programming experts use every day. These reasoning skill are then **developed** to a level where they can be used to construct and analyse working programs. Through extensive worked examples and exercises students first learn how to program then how to transcribe their programs. The rules for transcribing programs into Pascal are given at the end of each chapter. The chapters cover the following topics: **sequences** of **instructions** : **procedures** ; **procedures** with parameters; names and environments; counting repetitions; proofs by induction; choice; recursive **procedures** ; memory, input and assignment; input sequences and CRP, unlimited **repetition** ; repetitive formulas; termination, failure and searching; extended examples, printing a numeral; calculator, and printing a calendar; structures of values; and transcribing into other codes.
Subfile: C
Descriptors: programming
Identifiers: computer programming; proof of programs; worked examples; **procedures** ; counting repetitions; induction; choice; recursive **procedures** ; unlimited **repetition** ; repetitive formulas; termination
Class Codes: C4240 (Programming and algorithm theory); C6100 (Software techniques and systems); C6110 (Systems analysis and programming)

17/5/31 (Item 9 from file: 2)
DIALOG(R)File 2:INSPEC

01906009 INSPEC Abstract Number: C82032566
Title: Some sequential **selection** procedures **for good regression models**
Author(s): Tong-An Hsu; Deng-Yuan Huang
Author Affiliation: Nat. Central Univ., Chung-Li, Taiwan
Journal: Communications in Statistics - Theory and Methods vol.11, no.4 p.411-21
Publication Date: 1982 Country of Publication: USA
CODEN: CSTMDC ISSN: 0361-0926
Language: English Document Type: Journal Paper (JP)
Treatment: Theoretical (T)
Abstract: In the past decade a **number** of **fixed** sampling methods have been **developed** for selecting the 'best' or at least a 'good' subset of variable in regressions analysis. The authors derive a **sequential** selection **procedure** to select a subset of a random size including all good regression equations. Tables for an example are given. (6 Refs)
Subfile: C
Descriptors: statistical analysis
Identifiers: sequential selection; good regression models; fixed sampling methods; variable; regressions analysis; random size
Class Codes: C1140Z (Other and miscellaneous)

17/5/33 (Item 11 from file: 2)
DIALOG(R)File 2:INSPEC

01615255 INSPEC Abstract Number: C81002320
Title: Reduced program overhead for software- generated **dynamic memory refresh**
Author(s): Eggebrecht, L.C.
Author Affiliation: IBM Corp., Armonk, NY, USA
Journal: IBM Technical Disclosure Bulletin vol.23, no.2 p.468-9
Publication Date: July 1980 Country of Publication: USA
CODEN: IBMTAA ISSN: 0018-8689
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Dynamic memory requires periodic refresh to maintain its

information content. For data to be retained, each memory cell must be addressed within a time limit. Heretofore, a typical low cost refresh method is to periodically interrupt the system processor and have an interrupt service subroutine perform 128 row address sequential read. This software refresh **operation** is accomplished by using the low order seven address bits, and is **repeated** every two milliseconds. More specifically, this refresh **operation** is accomplished by 64 **sequential** 'dummy' POP **instructions** . Such periodic row address sequential read, however, takes up considerable processor time and adversely impacts its utilization. The author describes an improved interrupt refresh subroutine which reduces program overhead for such software- **generated** dynamic memory refresh. (0 Refs)

Subfile: C

Descriptors: data handling

Identifiers: reduced program overhead; software **generated** dynamic memory refresh; interrupt refresh subroutine

Class Codes: C6130 (Data handling techniques)

17/5/34 (Item 12 from file: 2)

DIALOG(R)File 2:INSPEC

00936525 INSPEC Abstract Number: C76018950

Title: Microprogram-control-circuits with compiler-language

Journal: Nachrichtentechnische Zeitschrift vol.29, no.5 p.365-9

Publication Date: May 1976 Country of Publication: West Germany

CODEN: NAZEAA ISSN: 0027-707X

Language: English Document Type: Journal Paper (JP)

Treatment: Applications (A)

Abstract: An explanation is given of the control-pulse representation useful during designing-process of digital-computing-machines. A programmable control-unit able to work with the **statements** of this language is introduced. In order to **generate** pulse- **repetition** and pulse-group- **repetition** the introduced control-unit concept is enlarged. This class of programmable control-unit is able to **generate** sequential logic-functions on order of control-program- **statements** . Costs of **sequential** and static logic-circuits are compared and enlargements of the control-circuit proposed. (5 Refs)

Subfile: C

Descriptors: program compilers

Identifiers: microprogram control circuit; compiler language; sequential logic functions

Class Codes: C6150C (Compilers, interpreters and other processors)